

SiWaSim

Generated by Doxygen 1.9.3

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Configuration Class Reference	5
3.1.1 Constructor & Destructor Documentation	6
3.1.1.1 Configuration()	6
3.1.1.2 ~Configuration()	6
3.1.2 Member Function Documentation	6
3.1.2.1 loadConfiguration()	6
3.1.3 Member Data Documentation	7
3.1.3.1 a	7
3.1.3.2 addvol_ratio	7
3.1.3.3 b	7
3.1.3.4 c	7
3.1.3.5 calibrationReg	7
3.1.3.6 cellCharecteristic	7
3.1.3.7 cellMode	8
3.1.3.8 d	8
3.1.3.9 endVoltage	8
3.1.3.10 exc_voltage	8
3.1.3.11 freqAt100	8
3.1.3.12 initial_weight	8
3.1.3.13 inputChannel1	8
3.1.3.14 inputChannel2	9
3.1.3.15 inputChannel3	9
3.1.3.16 inputChannel4	9
3.1.3.17 load_weight	9
3.1.3.18 max_diff_voltage	9
3.1.3.19 speedAt100	9
3.1.3.20 startVoltage	9
3.1.3.21 systemType	10
3.2 CUBIC_FUNCTION Struct Reference	10
3.2.1 Member Data Documentation	10
3.2.1.1 a	10
3.2.1.2 b	10
3.2.1.3 c	10
3.2.1.4 d	11
3.3 CURVE Struct Reference	11
3.3.1 Member Data Documentation	11

3.3.1.1 fallTime	11
3.3.1.2 maxFlow	11
3.3.1.3 riseTime	12
3.3.1.4 startDelay	12
3.3.1.5 stopDelay	12
3.4 GPIO Class Reference	12
3.4.1 Constructor & Destructor Documentation	12
3.4.1.1 GPIO()	13
3.4.1.2 ~GPIO()	13
3.4.2 Member Function Documentation	13
3.4.2.1 readPin()	13
3.4.2.2 setPinMode()	13
3.4.2.3 setPWM()	13
3.4.2.4 writePin()	13
3.5 I2C Class Reference	14
3.5.1 Constructor & Destructor Documentation	14
3.5.1.1 I2C()	14
3.5.1.2 ~I2C()	14
3.5.2 Member Function Documentation	14
3.5.2.1 begin()	14
3.5.2.2 readData() [1/2]	14
3.5.2.3 readData() [2/2]	15
3.5.2.4 writeData() [1/2]	15
3.5.2.5 writeData() [2/2]	15
3.6 IABoard Class Reference	15
3.6.1 Constructor & Destructor Documentation	16
3.6.1.1 IABoard()	16
3.6.1.2 ~IABoard()	16
3.6.2 Member Function Documentation	16
3.6.2.1 detectBoard()	16
3.6.2.2 digitalRead() [1/2]	16
3.6.2.3 digitalRead() [2/2]	17
3.6.2.4 getAnalogCurOut()	17
3.6.2.5 getAnalogVolOut()	17
3.6.2.6 getBoardData() [1/2]	17
3.6.2.7 getBoardData() [2/2]	17
3.6.2.8 getDigitalRead()	17
3.6.2.9 getLED()	17
3.6.2.10 getOpenDrainDOUT() [1/2]	18
3.6.2.11 getOpenDrainDOUT() [2/2]	18
3.6.2.12 getOpenDrainPWM()	18
3.6.2.13 getTransistionType()	18

3.6.2.14 readAnalogCurln()	18
3.6.2.15 readAnalogVolIn()	19
3.6.2.16 readAnalogVolInPM()	19
3.6.2.17 readTransistions()	19
3.6.2.18 resetTransitions()	20
3.6.2.19 setAllLED()	20
3.6.2.20 setAllOFF()	20
3.6.2.21 setAnalogCurOut()	20
3.6.2.22 setAnalogVolOut()	20
3.6.2.23 setLED()	20
3.6.2.24 setOpenDrainDOUT()	21
3.6.2.25 setOpenDrainPWM()	21
3.6.2.26 setTransistionType()	21
3.7 MaterialFlow Class Reference	21
3.7.1 Constructor & Destructor Documentation	22
3.7.1.1 MaterialFlow() [1/2]	22
3.7.1.2 MaterialFlow() [2/2]	22
3.7.1.3 ~MaterialFlow()	22
3.7.2 Member Function Documentation	22
3.7.2.1 setFlowCurve()	22
3.7.2.2 setFlowType()	23
3.7.2.3 update()	23
3.8 Modbus Class Reference	23
3.8.1 Constructor & Destructor Documentation	23
3.8.1.1 Modbus()	23
3.8.1.2 ~Modbus()	23
3.8.2 Member Function Documentation	24
3.8.2.1 receiveResponse()	24
3.8.2.2 transmitRequest()	24
3.9 PCB Class Reference	24
3.9.1 Constructor & Destructor Documentation	25
3.9.1.1 PCB()	25
3.9.1.2 ~PCB()	25
3.9.2 Member Function Documentation	25
3.9.2.1 getBoardStatus()	25
3.9.2.2 getEXCVoltage()	25
3.9.2.3 getSENVoltage()	25
3.9.2.4 ledBusy()	25
3.9.2.5 ledFault()	26
3.9.2.6 ledReady()	26
3.9.2.7 reloadConfig()	26
3.9.2.8 setAllOff()	26

3.9.2.9 setCellAddvol()	26
3.9.2.10 setCellSubvol()	26
3.9.2.11 setEXTRASW1()	26
3.9.2.12 setEXTRASW2()	27
3.9.2.13 setImpedance()	27
3.9.2.14 setLoadcellDCVoltage()	27
3.9.2.15 setLoadcellVoltage()	27
3.9.2.16 setPOWERSW1()	27
3.9.2.17 setPOWERSW2()	27
3.9.2.18 setPWM()	27
3.9.2.19 setSENVoltage()	28
3.10 Simulator Class Reference	28
3.10.1 Constructor & Destructor Documentation	28
3.10.1.1 Simulator()	28
3.10.1.2 ~Simulator()	28
3.10.2 Member Function Documentation	29
3.10.2.1 bootupAnimation()	29
3.10.2.2 calibrateLCVoltage()	29
3.10.2.3 reloadConfig()	29
3.10.2.4 run()	29
3.10.2.5 runPassive()	29
3.10.2.6 setImpedance()	29
3.10.2.7 setVelocity()	29
3.10.2.8 setVelocityFRQ()	30
3.10.2.9 setVelocityPER()	30
3.10.2.10 setWeightKG()	30
3.10.2.11 setWeightPER()	31
3.10.2.12 testFunction()	31
3.11 SIWAREX Class Reference	31
3.11.1 Constructor & Destructor Documentation	31
3.11.1.1 SIWAREX()	31
3.11.1.2 ~SIWAREX()	31
3.11.2 Member Function Documentation	32
3.11.2.1 getLoadcellVoltage()	32
3.12 UART Class Reference	32
3.12.1 Constructor & Destructor Documentation	32
3.12.1.1 UART()	32
3.12.1.2 ~UART()	32
3.12.2 Member Function Documentation	32
3.12.2.1 begin()	33
3.12.2.2 receiveMSG()	33
3.12.2.3 transmitMSG()	33

4 File Documentation	35
4.1 F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.cpp File Reference	35
4.2 F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.hpp File Reference	35
4.2.1 Macro Definition Documentation	36
4.2.1.1 ADDVOL_CHANNEL	36
4.2.1.2 CELL_DC	36
4.2.1.3 CONFIG_PATH	36
4.2.1.4 EXC_IN	37
4.2.1.5 I2C_ADDRESS	37
4.2.1.6 I2C_DEVICE	37
4.2.1.7 PIN_EXTRASW1	37
4.2.1.8 PIN_EXTRASW2	37
4.2.1.9 PIN_IMPEDANCE1	37
4.2.1.10 PIN_IMPEDANCE2	37
4.2.1.11 PIN_LED_BUSY	37
4.2.1.12 PIN_LED_FAULT	38
4.2.1.13 PIN_LED_READY	38
4.2.1.14 PIN_POWERSW1	38
4.2.1.15 PIN_POWERSW2	38
4.2.1.16 PWM_PIN	38
4.2.1.17 SEN_IN	38
4.2.1.18 SEN_OUT	38
4.2.1.19 SIWAREX_ADDRESS	38
4.2.1.20 SUBVOL_CHANNEL	39
4.2.2 Typedef Documentation	39
4.2.2.1 json	39
4.2.3 Enumeration Type Documentation	39
4.2.3.1 IMPEDANCE	39
4.2.3.2 LoadCellMode	39
4.2.3.3 MATERIAL_FLOW	40
4.2.3.4 RUN_MODE	40
4.2.3.5 SYSTEM_TYPE	40
4.3 Configuration.hpp	40
4.4 F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.cpp File Reference	42
4.5 F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.hpp File Reference	42
4.6 GPIO.hpp	42
4.7 F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.cpp File Reference	43
4.8 F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.hpp File Reference	43
4.9 I2C.hpp	43
4.10 F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.cpp File Reference	44
4.11 F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.hpp File Reference	44
4.11.1 Enumeration Type Documentation	44

4.11.1.1 TRANSITION	44
4.12 IABoard.hpp	45
4.13 F:/GITHUB/SiWaSIM-PiSoftware/src/main.cpp File Reference	46
4.13.1 Function Documentation	46
4.13.1.1 main()	46
4.14 F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.cpp File Reference	46
4.15 F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.hpp File Reference	47
4.16 MaterialFlow.hpp	47
4.17 F:/GITHUB/SiWaSIM-PiSoftware/src/Modbus.cpp File Reference	47
4.18 F:/GITHUB/SiWaSIM-PiSoftware/src/Modbus.hpp File Reference	48
4.19 Modbus.hpp	48
4.20 F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.cpp File Reference	48
4.21 F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.hpp File Reference	48
4.22 PCB.hpp	49
4.23 F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.cpp File Reference	49
4.24 F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.hpp File Reference	49
4.25 Simulator.hpp	50
4.26 F:/GITHUB/SiWaSIM-PiSoftware/src/SIWAREX.cpp File Reference	50
4.27 F:/GITHUB/SiWaSIM-PiSoftware/src/SIWAREX.hpp File Reference	50
4.28 SIWAREX.hpp	51
4.29 F:/GITHUB/SiWaSIM-PiSoftware/src/SIWAREX_REGISTER.hpp File Reference	51
4.29.1 Macro Definition Documentation	51
4.29.1.1 LOADCELL_VOLTAGE	51
4.30 SIWAREX_REGISTER.hpp	51
4.31 F:/GITHUB/SiWaSIM-PiSoftware/src/UART.cpp File Reference	51
4.32 F:/GITHUB/SiWaSIM-PiSoftware/src/UART.hpp File Reference	52
4.33 UART.hpp	52
4.34 F:/GITHUB/SiWaSIM-PiSoftware/src/utility.cpp File Reference	52
4.34.1 Function Documentation	53
4.34.1.1 bytesToFloat() [1/2]	53
4.34.1.2 bytesToFloat() [2/2]	53
4.34.1.3 calculateAverage()	53
4.34.1.4 calculateCubic()	53
4.34.1.5 calculateCubicDeriv()	54
4.34.1.6 constrainMax()	54
4.34.1.7 constrainMin()	55
4.34.1.8 constrainMinMax()	55
4.34.1.9 cubicRegression()	55
4.34.1.10 delay()	56
4.34.1.11 linearRegression()	56
4.34.1.12 solveCubicForVoltage()	56
4.35 F:/GITHUB/SiWaSIM-PiSoftware/src/utility.hpp File Reference	57

4.35.1 Function Documentation	57
4.35.1.1 bytesToFloat() [1/2]	58
4.35.1.2 bytesToFloat() [2/2]	58
4.35.1.3 calculateAverage()	58
4.35.1.4 calculateCubic()	58
4.35.1.5 calculateCubicDeriv()	59
4.35.1.6 constrainMax()	59
4.35.1.7 constrainMin()	60
4.35.1.8 constrainMinMax()	60
4.35.1.9 cubicRegression()	60
4.35.1.10 delay()	61
4.35.1.11 linearRegression()	61
4.35.1.12 solveCubicForVoltage()	61
4.36 utility.hpp	62
Index	63

Chapter 1

Class Index

1.1 Class List

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

Configuration	5
CUBIC_FUNCTION	10
CURVE	11
GPIO	12
I2C	14
IABoard	15
MaterialFlow	21
Modbus	23
PCB	24
Simulator	28
SIWAREX	31
UART	32

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

F:/GITHUB/SiWaSIM-PiSoftware/src/ Configuration.cpp	35
F:/GITHUB/SiWaSIM-PiSoftware/src/ Configuration.hpp	35
F:/GITHUB/SiWaSIM-PiSoftware/src/ GPIO.cpp	42
F:/GITHUB/SiWaSIM-PiSoftware/src/ GPIO.hpp	42
F:/GITHUB/SiWaSIM-PiSoftware/src/ I2C.cpp	43
F:/GITHUB/SiWaSIM-PiSoftware/src/ I2C.hpp	43
F:/GITHUB/SiWaSIM-PiSoftware/src/ IABoard.cpp	44
F:/GITHUB/SiWaSIM-PiSoftware/src/ IABoard.hpp	44
F:/GITHUB/SiWaSIM-PiSoftware/src/ main.cpp	46
F:/GITHUB/SiWaSIM-PiSoftware/src/ MaterialFlow.cpp	46
F:/GITHUB/SiWaSIM-PiSoftware/src/ MaterialFlow.hpp	47
F:/GITHUB/SiWaSIM-PiSoftware/src/ Modbus.cpp	47
F:/GITHUB/SiWaSIM-PiSoftware/src/ Modbus.hpp	48
F:/GITHUB/SiWaSIM-PiSoftware/src/ PCB.cpp	48
F:/GITHUB/SiWaSIM-PiSoftware/src/ PCB.hpp	48
F:/GITHUB/SiWaSIM-PiSoftware/src/ Simulator.cpp	49
F:/GITHUB/SiWaSIM-PiSoftware/src/ Simulator.hpp	49
F:/GITHUB/SiWaSIM-PiSoftware/src/ SIWAREX.cpp	50
F:/GITHUB/SiWaSIM-PiSoftware/src/ SIWAREX.hpp	50
F:/GITHUB/SiWaSIM-PiSoftware/src/ SIWAREX_REGISTER.hpp	51
F:/GITHUB/SiWaSIM-PiSoftware/src/ UART.cpp	51
F:/GITHUB/SiWaSIM-PiSoftware/src/ UART.hpp	52
F:/GITHUB/SiWaSIM-PiSoftware/src/ utility.cpp	52
F:/GITHUB/SiWaSIM-PiSoftware/src/ utility.hpp	57

Chapter 3

Class Documentation

3.1 Configuration Class Reference

```
#include <Configuration.hpp>
```

Public Member Functions

- [Configuration](#) (std::string path)
- [~Configuration](#) ()
- void [loadConfiguration](#) ()

Public Attributes

- [LoadCellMode](#) [cellMode](#) = [LoadCellMode::NORMAL](#)
Loadcell mode to be simulated.
- [SYSTEM_TYPE](#) [systemType](#) = [SYSTEM_TYPE::DOSING_SCALE](#)
System type to be simulated.
- float [exc_voltage](#) = 10.f
Nominal EXC voltage ouputted by the [SIWAREX](#) module.
- float [load_weight](#) = 20.f
Nominal Load Weight of the cell in kg.
- float [initial_weight](#) = 10.f
Initial weight (for manual / non-auto mode)
- float [addVol_ratio](#) = 500
Inverted OpAmp gain (e.g.: At 10V Aout the added / subtracted voltage is 20mV --> ratio = 10V / 20mV = 500)
- float [max_diff_voltage](#) = 40
Maximum Differential Voltage of SIG+-.
- float [cellCharecteristic](#) = 4
Characteristic in mV/V.
- float [speedAt100](#) = 5
Belt velocity in m/s at 100% speed.
- float [freqAt100](#) = 10000
Belt encoder frequency at 100% speed.
- float [startVoltage](#) = 2

- float `endVoltage` = 9
- float `a`
- float `b`
- float `c`
- float `d`
- `CUBIC_FUNCTION` `calibrationReg`
- `MATERIAL_FLOW` `inputChannel1` = `MATERIAL_FLOW::EMPTY`
- `MATERIAL_FLOW` `inputChannel2` = `MATERIAL_FLOW::FINE`
- `MATERIAL_FLOW` `inputChannel3` = `MATERIAL_FLOW::COARSE`
- `MATERIAL_FLOW` `inputChannel4` = `MATERIAL_FLOW::XCOARSE`

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Configuration()

```
Configuration::Configuration (
    std::string path )
```

Creates a new configuration that stores all configuration settings needed for the [Simulator](#). IMPORTANT: Should only be created once, since there is only one valid configuration for the simulator!

Parameters

<i>path</i>	The path to the configuration file on the filesystem
-------------	--

3.1.1.2 ~Configuration()

```
Configuration::~~Configuration ( )
```

3.1.2 Member Function Documentation

3.1.2.1 loadConfiguration()

```
void Configuration::loadConfiguration ( )
```

Loads a configuration file from the file system (specified by path in [Configuration\(std::string path\)](#)) and parses all settings to their respective variables

3.1.3 Member Data Documentation

3.1.3.1 a

```
float Configuration::a
```

3.1.3.2 addvol_ratio

```
float Configuration::addvol_ratio = 500
```

Inverted OpAmp gain (e.g.: At 10V Aout the added / subtracted voltage is 20mV --> ratio = 10V / 20mV = 500)

3.1.3.3 b

```
float Configuration::b
```

3.1.3.4 c

```
float Configuration::c
```

3.1.3.5 calibrationReg

```
CUBIC_FUNCTION Configuration::calibrationReg
```

3.1.3.6 cellCharecteristic

```
float Configuration::cellCharecteristic = 4
```

Characteristic in mV/V.

3.1.3.7 cellMode

`LoadCellMode` Configuration::cellMode = `LoadCellMode::NORMAL`

Loadcell mode to be simulated.

3.1.3.8 d

`float` Configuration::d

3.1.3.9 endVoltage

`float` Configuration::endVoltage = 9

3.1.3.10 exc_voltage

`float` Configuration::exc_voltage = 10.f

Nominal EXC voltage ouputted by the `SIWAREX` module.

3.1.3.11 freqAt100

`float` Configuration::freqAt100 = 10000

Belt encoder frequency at 100% speed.

3.1.3.12 initial_weight

`float` Configuration::initial_weight = 10.f

Initial weight (for manual / non-auto mode)

3.1.3.13 inputChannel1

`MATERIAL_FLOW` Configuration::inputChannel1 = `MATERIAL_FLOW::EMPTY`

3.1.3.14 inputChannel2

`MATERIAL_FLOW` Configuration::inputChannel2 = `MATERIAL_FLOW::FINE`

3.1.3.15 inputChannel3

`MATERIAL_FLOW` Configuration::inputChannel3 = `MATERIAL_FLOW::COARSE`

3.1.3.16 inputChannel4

`MATERIAL_FLOW` Configuration::inputChannel4 = `MATERIAL_FLOW::XCOARSE`

3.1.3.17 load_weight

`float` Configuration::load_weight = 20.f

Nominal Load Weight of the cell in kg.

3.1.3.18 max_diff_voltage

`float` Configuration::max_diff_voltage = 40

Maximum Differential Voltage of SIG+/-.

3.1.3.19 speedAt100

`float` Configuration::speedAt100 = 5

Belt velocity in m/s at 100% speed.

3.1.3.20 startVoltage

`float` Configuration::startVoltage = 2

3.1.3.21 systemType

```
SYSTEM_TYPE Configuration::systemType = SYSTEM_TYPE::DOSING_SCALE
```

System type to be simulated.

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/[Configuration.hpp](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/[Configuration.cpp](#)

3.2 CUBIC_FUNCTION Struct Reference

```
#include <Configuration.hpp>
```

Public Attributes

- float [a](#)
- float [b](#)
- float [c](#)
- float [d](#)

3.2.1 Member Data Documentation

3.2.1.1 a

```
float CUBIC_FUNCTION::a
```

3.2.1.2 b

```
float CUBIC_FUNCTION::b
```

3.2.1.3 c

```
float CUBIC_FUNCTION::c
```

3.2.1.4 d

```
float CUBIC_FUNCTION::d
```

The documentation for this struct was generated from the following file:

- F:/GITHUB/SiWaSIM-PiSoftware/src/[Configuration.hpp](#)

3.3 CURVE Struct Reference

```
#include <MaterialFlow.hpp>
```

Public Attributes

- float [startDelay](#) = 0
Delay from input high to flow increase start in seconds.
- float [stopDelay](#) = 0
Delay from input low to flow decrease start in seconds.
- float [riseTime](#) = 1
Time it takes the flow to reach its maximum in seconds.
- float [fallTime](#) = 1
Time it takes the flow to reach zero in seconds.
- float [maxFlow](#) = 1
Maximal flow after rise time in kg/s.

3.3.1 Member Data Documentation

3.3.1.1 fallTime

```
float CURVE::fallTime = 1
```

Time it takes the flow to reach zero in seconds.

3.3.1.2 maxFlow

```
float CURVE::maxFlow = 1
```

Maximal flow after rise time in kg/s.

3.3.1.3 riseTime

```
float CURVE::riseTime = 1
```

Time it takes the flow to reach its maximum in seconds.

3.3.1.4 startDelay

```
float CURVE::startDelay = 0
```

Delay from input high to flow increase start in seconds.

3.3.1.5 stopDelay

```
float CURVE::stopDelay = 0
```

Delay from input low to flow decrease start in seconds.

The documentation for this struct was generated from the following file:

- <F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.hpp>

3.4 GPIO Class Reference

```
#include <GPIO.hpp>
```

Public Member Functions

- [GPIO](#) ()
- [~GPIO](#) ()
- void [setPWM](#) (int pin, float dutyCycle, float frequency)
- void [setPinMode](#) (uint8_t pin, uint8_t mode)
- void [writePin](#) (uint8_t pin, bool state)
- bool [readPin](#) (uint8_t pin)

3.4.1 Constructor & Destructor Documentation

3.4.1.1 GPIO()

```
GPIO::GPIO ( )
```

3.4.1.2 ~GPIO()

```
GPIO::~~GPIO ( )
```

3.4.2 Member Function Documentation

3.4.2.1 readPin()

```
bool GPIO::readPin (
    uint8_t pin )
```

3.4.2.2 setPinMode()

```
void GPIO::setPinMode (
    uint8_t pin,
    uint8_t mode )
```

3.4.2.3 setPWM()

```
void GPIO::setPWM (
    int pin,
    float dutyCycle,
    float frequency )
```

3.4.2.4 writePin()

```
void GPIO::writePin (
    uint8_t pin,
    bool state )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/[GPIO.hpp](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/[GPIO.cpp](#)

3.5 I2C Class Reference

```
#include <I2C.hpp>
```

Public Member Functions

- [I2C](#) (std::string dev, uint16_t address)
- [~I2C](#) ()
- bool [begin](#) ()
- bool [writeData](#) (uint8_t data)
- bool [writeData](#) (uint8_t *data, uint8_t length)
- bool [readData](#) (uint8_t *data, uint8_t length)
- uint8_t [readData](#) ()

3.5.1 Constructor & Destructor Documentation

3.5.1.1 I2C()

```
I2C::I2C (
    std::string dev,
    uint16_t address )
```

3.5.1.2 ~I2C()

```
I2C::~I2C ( )
```

3.5.2 Member Function Documentation

3.5.2.1 begin()

```
bool I2C::begin ( )
```

3.5.2.2 readData() [1/2]

```
uint8_t I2C::readData ( )
```


3.5.2.3 readData() [2/2]

```
bool I2C::readData (
    uint8_t * data,
    uint8_t length )
```

3.5.2.4 writeData() [1/2]

```
bool I2C::writeData (
    uint8_t * data,
    uint8_t length )
```

3.5.2.5 writeData() [2/2]

```
bool I2C::writeData (
    uint8_t data )
```

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

- [F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.hpp](#)
- [F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.cpp](#)

3.6 IABoard Class Reference

```
#include <IABoard.hpp>
```

Public Member Functions

- [IABoard \(\)](#)
- [~IABoard \(\)](#)
- bool [detectBoard \(\)](#)
- uint8_t [digitalRead \(\)](#)
- bool [digitalRead \(uint8_t channel\)](#)
- bool [getDigitalRead \(uint8_t channel\)](#)
- uint16_t [readTransistions \(uint8_t channel\)](#)
- [TRANSITION](#) [getTransistionType \(uint8_t channel\)](#)
- void [setTransistionType \(uint8_t channel, TRANSITION tran\)](#)
- void [resetTransitions \(uint8_t channel\)](#)
- float [getAnalogVolOut \(uint8_t channel\)](#)
- void [setAnalogVolOut \(uint8_t channel, float voltage\)](#)
- float [getAnalogCurOut \(uint8_t channel\)](#)
- void [setAnalogCurOut \(uint8_t channel, float current\)](#)
- float [getOpenDrainPWM \(uint8_t channel\)](#)
- void [setOpenDrainPWM \(uint8_t channel, float dutyCycle\)](#)
- uint8_t [getOpenDrainDOUT \(\)](#)

- bool [getOpenDrainDOUT](#) (uint8_t channel)
- void [setOpenDrainDOUT](#) (uint8_t channel, bool value)
- bool [getLED](#) (uint8_t channel)
- void [setLED](#) (uint8_t channel, bool value)
- void [setAllLED](#) (bool value)
- float [readAnalogVolln](#) (uint8_t channel)
- float [readAnalogVollnPM](#) (uint8_t channel)
- float [readAnalogCurln](#) (uint8_t channel)
- void [getBoardData](#) ()
- void [getBoardData](#) (uint8_t *temp, float *rail24, float *rail5)
- void [setAllOFF](#) ()

3.6.1 Constructor & Destructor Documentation

3.6.1.1 IABoard()

```
IABoard::IABoard ( )
```

3.6.1.2 ~IABoard()

```
IABoard::~~IABoard ( )
```

3.6.2 Member Function Documentation

3.6.2.1 detectBoard()

```
bool IABoard::detectBoard ( )
```

3.6.2.2 digitalRead() [1/2]

```
uint8_t IABoard::digitalRead ( )
```

3.6.2.3 digitalRead() [2/2]

```
bool IABoard::digitalRead (
    uint8_t channel )
```

3.6.2.4 getAnalogCurOut()

```
float IABoard::getAnalogCurOut (
    uint8_t channel )
```

3.6.2.5 getAnalogVolOut()

```
float IABoard::getAnalogVolOut (
    uint8_t channel )
```

3.6.2.6 getBoardData() [1/2]

```
void IABoard::getBoardData ( )
```

Receives the board data through the command. Board data includes temperature, 24V input rail voltage and 5V rail voltage

3.6.2.7 getBoardData() [2/2]

```
void IABoard::getBoardData (
    uint8_t * temp,
    float * rail24,
    float * rail5 )
```

3.6.2.8 getDigitalRead()

```
bool IABoard::getDigitalRead (
    uint8_t channel )
```

3.6.2.9 getLED()

```
bool IABoard::getLED (
    uint8_t channel )
```

Gets the current state of one of the on board LEDs

Parameters

<i>channel</i>	The LED to be read (1 - 4)
----------------	----------------------------

Returns

Returns the state of the LED (0 = OFF, 1 = ON)

3.6.2.10 getOpenDrainDOUT() [1/2]

```
uint8_t IABoard::getOpenDrainDOUT ( )
```

3.6.2.11 getOpenDrainDOUT() [2/2]

```
bool IABoard::getOpenDrainDOUT (
    uint8_t channel )
```

3.6.2.12 getOpenDrainPWM()

```
float IABoard::getOpenDrainPWM (
    uint8_t channel )
```

3.6.2.13 getTransistionType()

```
TRANSITION IABoard::getTransistionType (
    uint8_t channel )
```

3.6.2.14 readAnalogCurIn()

```
float IABoard::readAnalogCurIn (
    uint8_t channel )
```

Reads the Analog Input Current of a channel

Parameters

<i>channel</i>	The channel as marked on the IABoard-PCB (1 - 4)
----------------	--

Returns

Returns the measured current in mA

3.6.2.15 readAnalogVolIn()

```
float IABoard::readAnalogVolIn (
    uint8_t channel )
```

Reads the Analog Input Voltage of a channel if the jumper is not set

Parameters

<i>channel</i>	The channel as marked on the IABoard-PCB (1 - 4)
----------------	--

Returns

Returns the measured voltage in Volts from 0V to 10V

3.6.2.16 readAnalogVolInPM()

```
float IABoard::readAnalogVolInPM (
    uint8_t channel )
```

Reads the Analog Input Voltage of a channel if the jumper is set to measure negative voltages

Parameters

<i>channel</i>	The channel as marked on the IABoard-PCB (1 - 4)
----------------	--

Returns

Returns the measured voltage in Volts from -10V to 10V

3.6.2.17 readTransistions()

```
uint16_t IABoard::readTransistions (
    uint8_t channel )
```

3.6.2.18 resetTransitions()

```
void IABoard::resetTransitions (
    uint8_t channel )
```

3.6.2.19 setAllLED()

```
void IABoard::setAllLED (
    bool value )
```

Sets all IABoard-LEDs to the same state

Parameters

<i>value</i>	The wanted state of all the LEDs (0 = OFF, 1 = ON)
--------------	--

3.6.2.20 setAllOFF()

```
void IABoard::setAllOFF ( )
```

Sets all digital and analog outputs to OFF / 0V

3.6.2.21 setAnalogCurOut()

```
void IABoard::setAnalogCurOut (
    uint8_t channel,
    float current )
```

3.6.2.22 setAnalogVolOut()

```
void IABoard::setAnalogVolOut (
    uint8_t channel,
    float voltage )
```

3.6.2.23 setLED()

```
void IABoard::setLED (
    uint8_t channel,
    bool value )
```

Sets on of the four on board LEDs to a certain state

Parameters

<i>channel</i>	The LED to be toggled (1 - 4)
<i>value</i>	The wanted state of the LED (0 = OFF, 1 = ON)

3.6.2.24 setOpenDrainDOUT()

```
void IABoard::setOpenDrainDOUT (
    uint8_t channel,
    bool value )
```

Sets on of the four digital outputs

Parameters

<i>channel</i>	The Open Drain Pin to be toggled (1 - 4)
<i>value</i>	The wanted state of the channel

3.6.2.25 setOpenDrainPWM()

```
void IABoard::setOpenDrainPWM (
    uint8_t channel,
    float dutyCycle )
```

3.6.2.26 setTransistionType()

```
void IABoard::setTransistionType (
    uint8_t channel,
    TRANSITION tran )
```

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

- <F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.hpp>
- <F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.cpp>

3.7 MaterialFlow Class Reference

```
#include <MaterialFlow.hpp>
```

Public Member Functions

- [MaterialFlow](#) (uint8_t channel)
- [MaterialFlow](#) (uint8_t channel, [MATERIAL_FLOW](#) flowType)
- [~MaterialFlow](#) ()
- void [setFlowCurve](#) ([CURVE](#) curve)
- void [setFlowType](#) ([MATERIAL_FLOW](#) flowType)
- float [update](#) (float *currentWeight, float dt, bool pinState)

3.7.1 Constructor & Destructor Documentation

3.7.1.1 [MaterialFlow\(\)](#) [1/2]

```
MaterialFlow::MaterialFlow (
    uint8_t channel )
```

3.7.1.2 [MaterialFlow\(\)](#) [2/2]

```
MaterialFlow::MaterialFlow (
    uint8_t channel,
    MATERIAL\_FLOW flowType )
```

3.7.1.3 [~MaterialFlow\(\)](#)

```
MaterialFlow::~MaterialFlow ( )
```

3.7.2 Member Function Documentation

3.7.2.1 [setFlowCurve\(\)](#)

```
void MaterialFlow::setFlowCurve (
    CURVE curve )
```


3.7.2.2 setFlowType()

```
void MaterialFlow::setFlowType (
    MATERIAL_FLOW flowType )
```

3.7.2.3 update()

```
float MaterialFlow::update (
    float * currentWeight,
    float dt,
    bool pinState )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/[MaterialFlow.hpp](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/[MaterialFlow.cpp](#)

3.8 Modbus Class Reference

```
#include <Modbus.hpp>
```

Public Member Functions

- [Modbus](#) ()
- [~Modbus](#) ()
- void [transmitRequest](#) (uint16_t startRegister, uint16_t length)
- std::vector< uint8_t > [receiveResponse](#) ()

3.8.1 Constructor & Destructor Documentation

3.8.1.1 Modbus()

```
Modbus::Modbus ( )
```

3.8.1.2 ~Modbus()

```
Modbus::~~Modbus ( )
```

3.8.2 Member Function Documentation

3.8.2.1 receiveResponse()

```
std::vector< uint8_t > Modbus::receiveResponse ( )
```

3.8.2.2 transmitRequest()

```
void Modbus::transmitRequest (
    uint16_t startRegister,
    uint16_t length )
```

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

- [F:/GITHUB/SiWaSIM-PiSoftware/src/Modbus.hpp](#)
- [F:/GITHUB/SiWaSIM-PiSoftware/src/Modbus.cpp](#)

3.9 PCB Class Reference

```
#include <PCB.hpp>
```

Public Member Functions

- [PCB](#) ([Configuration](#) *config)
- [~PCB](#) ()
- void [ledFault](#) (bool state)
- void [ledBusy](#) (bool state)
- void [ledReady](#) (bool state)
- void [setImpedance](#) ([IMPEDANCE](#) impedance)
- void [setEXTRASW1](#) (bool state)
- void [setEXTRASW2](#) (bool state)
- void [setPOWERSW1](#) (bool state)
- void [setPOWERSW2](#) (bool state)
- void [setLoadcellVoltage](#) (float voltage)
- void [setLoadcellDCVoltage](#) (float voltage)
- void [setCellAddvol](#) (float voltage)
- void [setCellSubvol](#) (float voltage)
- void [setSENVoltage](#) (float voltage)
- float [getEXCVoltage](#) ()
- float [getSENVoltage](#) ()
- void [setPWM](#) (float frequency, float dutyCycle)
- void [getBoardStatus](#) ()
- void [reloadConfig](#) ()
- void [setAllOff](#) ()

3.9.1 Constructor & Destructor Documentation

3.9.1.1 PCB()

```
PCB::PCB (
    Configuration * config )
```

3.9.1.2 ~PCB()

```
PCB::~~PCB ( )
```

3.9.2 Member Function Documentation

3.9.2.1 getBoardStatus()

```
void PCB::getBoardStatus ( )
```

3.9.2.2 getEXCVoltage()

```
float PCB::getEXCVoltage ( )
```

3.9.2.3 getSENVoltage()

```
float PCB::getSENVoltage ( )
```

3.9.2.4 ledBusy()

```
void PCB::ledBusy (
    bool state )
```

3.9.2.5 ledFault()

```
void PCB::ledFault (
    bool state )
```

3.9.2.6 ledReady()

```
void PCB::ledReady (
    bool state )
```

3.9.2.7 reloadConfig()

```
void PCB::reloadConfig ( )
```

3.9.2.8 setAllOff()

```
void PCB::setAllOff ( )
```

3.9.2.9 setCellAddvol()

```
void PCB::setCellAddvol (
    float voltage )
```

3.9.2.10 setCellSubvol()

```
void PCB::setCellSubvol (
    float voltage )
```

3.9.2.11 setEXTRASW1()

```
void PCB::setEXTRASW1 (
    bool state )
```

3.9.2.12 setEXTRASW2()

```
void PCB::setEXTRASW2 (
    bool state )
```

3.9.2.13 setImpedance()

```
void PCB::setImpedance (
    IMPEDANCE impedance )
```

3.9.2.14 setLoadcellDCVoltage()

```
void PCB::setLoadcellDCVoltage (
    float voltage )
```

3.9.2.15 setLoadcellVoltage()

```
void PCB::setLoadcellVoltage (
    float voltage )
```

3.9.2.16 setPOWERSW1()

```
void PCB::setPOWERSW1 (
    bool state )
```

3.9.2.17 setPOWERSW2()

```
void PCB::setPOWERSW2 (
    bool state )
```

3.9.2.18 setPWM()

```
void PCB::setPWM (
    float frequency,
    float dutyCycle )
```

3.9.2.19 setSENVoltage()

```
void PCB::setSENVoltage (
    float voltage )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/[PCB.hpp](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/[PCB.cpp](#)

3.10 Simulator Class Reference

```
#include <Simulator.hpp>
```

Public Member Functions

- [Simulator](#) ()
- [~Simulator](#) ()
- void [setWeightPER](#) (float percentage)
- void [setWeightKG](#) (float kg)
- void [setVelocity](#) (float meterspersecond)
- void [setVelocityPER](#) (float percentage)
- void [setVelocityFRQ](#) (float frequency)
- void [setImpedance](#) ([IMPEDANCE](#) impedance)
- void [bootupAnimation](#) ()
- void [reloadConfig](#) ()
- void [testFunction](#) ()
- float [run](#) ([RUN_MODE](#) runMode, float timestep, float *weight)
- float [runPassive](#) (float timestep, float *weight)
- void [calibrateLCVoltage](#) (bool autoCalib)

3.10.1 Constructor & Destructor Documentation

3.10.1.1 Simulator()

```
Simulator::Simulator ( )
```

3.10.1.2 ~Simulator()

```
Simulator::~~Simulator ( )
```

3.10.2 Member Function Documentation

3.10.2.1 bootupAnimation()

```
void Simulator::bootupAnimation ( )
```

Starts an animation with the on board LEDs

3.10.2.2 calibrateLCVoltage()

```
void Simulator::calibrateLCVoltage (
    bool autoCalib )
```

3.10.2.3 reloadConfig()

```
void Simulator::reloadConfig ( )
```

Reloads the configuration from the disk and stores the settings

3.10.2.4 run()

```
float Simulator::run (
    RUN_MODE runMode,
    float timestep,
    float * weight )
```

3.10.2.5 runPassive()

```
float Simulator::runPassive (
    float timestep,
    float * weight )
```

3.10.2.6 setImpedance()

```
void Simulator::setImpedance (
    IMPEDANCE impedance )
```

3.10.2.7 setVelocity()

```
void Simulator::setVelocity (
    float meterspersecond )
```

Sets the simulated belt velocity in meters per second

Parameters

<i>meterspersecond</i>	Velocity in meters / second
------------------------	-----------------------------

3.10.2.8 setVelocityFRQ()

```
void Simulator::setVelocityFRQ (  
    float frequency )
```

Sets the PWM output to a certain frequency to represent belt movement

Parameters

<i>frequency</i>	The frequency of the PWM signal
------------------	---------------------------------

3.10.2.9 setVelocityPER()

```
void Simulator::setVelocityPER (  
    float percentage )
```

Sets the simulated belt velocity from 0 - 100% of the maximal speed

Parameters

<i>percentage</i>	Percentage of the maximal speed from 0 to 1
-------------------	---

3.10.2.10 setWeightKG()

```
void Simulator::setWeightKG (  
    float kg )
```

Set the output weight of the simulated load cell in kg

Parameters

<i>kg</i>	Output weight in kilograms
-----------	----------------------------

3.10.2.11 setWeightPER()

```
void Simulator::setWeightPER (
    float percentage )
```

Set the output weight as a percentage of the nominal load

Parameters

<i>percentage</i>	Percentage from 0 - 1 where 1 represents the nominal load as specified
-------------------	--

3.10.2.12 testFunction()

```
void Simulator::testFunction ( )
```

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

- [F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.hpp](#)
- [F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.cpp](#)

3.11 SIWAREX Class Reference

```
#include <SIWAREX.hpp>
```

Public Member Functions

- [SIWAREX \(\)](#)
- [~SIWAREX \(\)](#)
- float [getLoadcellVoltage \(\)](#)

3.11.1 Constructor & Destructor Documentation

3.11.1.1 SIWAREX()

```
SIWAREX::SIWAREX ( )
```

3.11.1.2 ~SIWAREX()

```
SIWAREX::~SIWAREX ( )
```

3.11.2 Member Function Documentation

3.11.2.1 getLoadcellVoltage()

```
float SIWAREX::getLoadcellVoltage ( )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/[SIWAREX.hpp](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/[SIWAREX.cpp](#)

3.12 UART Class Reference

```
#include <UART.hpp>
```

Public Member Functions

- [UART](#) ()
- [~UART](#) ()
- bool [begin](#) ()
- bool [transmitMSG](#) (uint8_t *msg, uint16_t length)
- std::vector< uint8_t > [receiveMSG](#) ()

3.12.1 Constructor & Destructor Documentation

3.12.1.1 UART()

```
UART::UART ( )
```

3.12.1.2 ~UART()

```
UART::~~UART ( )
```

3.12.2 Member Function Documentation

3.12.2.1 begin()

```
bool UART::begin ( )
```

3.12.2.2 receiveMSG()

```
std::vector< uint8_t > UART::receiveMSG ( )
```

3.12.2.3 transmitMSG()

```
bool UART::transmitMSG (
    uint8_t * msg,
    uint16_t length )
```

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

- [F:/GITHUB/SiWaSIM-PiSoftware/src/UART.hpp](#)
- [F:/GITHUB/SiWaSIM-PiSoftware/src/UART.cpp](#)

Chapter 4

File Documentation

4.1 F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.cpp File Reference

```
#include "Configuration.hpp"
```

4.2 F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.hpp File Reference

```
#include <string>
#include <iostream>
#include <fstream>
#include "nlohmann/json.hpp"
```

Classes

- struct [CUBIC_FUNCTION](#)
- class [Configuration](#)

Macros

- #define [CONFIG_PATH](#) "/home/siwasim/SiWaSIM-PiSoftware/Konfiguration/config.json"
- #define [I2C_ADDRESS](#) 0x50
- #define [I2C_DEVICE](#) "/dev/i2c-1"
- #define [SIWAREX_ADDRESS](#) 0x14
- #define [PIN_LED_READY](#) 23
- #define [PIN_LED_BUSY](#) 24
- #define [PIN_LED_FAULT](#) 25
- #define [PWM_PIN](#) 13
- #define [PIN_POWERSW1](#) 4
- #define [PIN_POWERSW2](#) 26
- #define [PIN_IMPEDANCE1](#) 5
- #define [PIN_IMPEDANCE2](#) 6

- `#define PIN_EXTRASW1 27`
- `#define PIN_EXTRASW2 22`
- `#define ADDVOL_CHANNEL 2`
- `#define SUBVOL_CHANNEL 3`
- `#define CELL_DC 1`
- `#define SEN_OUT 4`
- `#define EXC_IN 1`
- `#define SEN_IN 2`

Typedefs

- using `json` = `nlohmann::json`

Enumerations

- enum `LoadCellMode` { `NORMAL` = 0x00 , `OVERLOAD` = 0x01 , `INVERTED` = 0x02 }
- enum `IMPEDANCE` { `OPEN` = 0x00 , `NOMINAL` = 0x01 , `SHORT` = 0x02 }
- enum `SYSTEM_TYPE` { `DOSING_SCALE` = 0x01 , `BELT_SCALE` = 0x02 }
- enum `MATERIAL_FLOW` {
 `NONE` = 0x00 , `EMPTY` , `FINE` , `COARSE` ,
 `XCOARSE` }
- enum `RUN_MODE` {
 `AUTO` , `PASSIVE` , `MANUAL` , `IDLE` ,
 `OFF` }

4.2.1 Macro Definition Documentation

4.2.1.1 ADDVOL_CHANNEL

```
#define ADDVOL_CHANNEL 2
```

4.2.1.2 CELL_DC

```
#define CELL_DC 1
```

4.2.1.3 CONFIG_PATH

```
#define CONFIG_PATH "/home/siwasim/SiWaSIM-PiSoftware/Konfiguration/config.json"
```

4.2.1.4 EXC_IN

```
#define EXC_IN 1
```

4.2.1.5 I2C_ADDRESS

```
#define I2C_ADDRESS 0x50
```

4.2.1.6 I2C_DEVICE

```
#define I2C_DEVICE "/dev/i2c-1"
```

4.2.1.7 PIN_EXTRASW1

```
#define PIN_EXTRASW1 27
```

4.2.1.8 PIN_EXTRASW2

```
#define PIN_EXTRASW2 22
```

4.2.1.9 PIN_IMPEDANCE1

```
#define PIN_IMPEDANCE1 5
```

4.2.1.10 PIN_IMPEDANCE2

```
#define PIN_IMPEDANCE2 6
```

4.2.1.11 PIN_LED_BUSY

```
#define PIN_LED_BUSY 24
```

4.2.1.12 PIN_LED_FAULT

```
#define PIN_LED_FAULT 25
```

4.2.1.13 PIN_LED_READY

```
#define PIN_LED_READY 23
```

4.2.1.14 PIN_POWERSW1

```
#define PIN_POWERSW1 4
```

4.2.1.15 PIN_POWERSW2

```
#define PIN_POWERSW2 26
```

4.2.1.16 PWM_PIN

```
#define PWM_PIN 13
```

4.2.1.17 SEN_IN

```
#define SEN_IN 2
```

4.2.1.18 SEN_OUT

```
#define SEN_OUT 4
```

4.2.1.19 SIWAREX_ADDRESS

```
#define SIWAREX_ADDRESS 0x14
```


4.2.1.20 SUBVOL_CHANNEL

```
#define SUBVOL_CHANNEL 3
```

4.2.2 Typedef Documentation

4.2.2.1 json

```
using json = nlohmann::json
```

4.2.3 Enumeration Type Documentation

4.2.3.1 IMPEDANCE

```
enum IMPEDANCE
```

Types of impedances of the load cell that can be simulated. Is equivalent with the impedance between EXC+ and EXC-

Enumerator

OPEN	Open circuit, high impedance.
NOMINAL	Nominal impedance of approx. 350 ohms.
SHORT	Short circuit, approx. zero impedance.

4.2.3.2 LoadCellMode

```
enum LoadCellMode
```

Enumerator

NORMAL	Positive differential voltage from 0 - 100% nominal load.
OVERLOAD	Positive differential voltage from 0 - 120% nominal load.
INVERTED	Negative differential voltage from 0 - 100% nominal load.

4.2.3.3 MATERIAL_FLOW

enum [MATERIAL_FLOW](#)

Types of different material flows

Enumerator

NONE	
EMPTY	
FINE	
COARSE	
XCOARSE	

4.2.3.4 RUN_MODE

enum [RUN_MODE](#)

Enumerator

AUTO	
PASSIVE	
MANUAL	
IDLE	
OFF	

4.2.3.5 SYSTEM_TYPE

enum [SYSTEM_TYPE](#)

Type of the system represented by the simulator

Enumerator

DOSING_SCALE	Dosing Scale.
BELT_SCALE	Belt Scale.

4.3 Configuration.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #include <string>
3 #include <iostream>
```

```

4 #include <fstream>
5 #include "nlohmann/json.hpp"
6
7 using json = nlohmann::json;
8
9 #define CONFIG_PATH "/home/siwasim/SiWaSIM-PiSoftware/Konfiguration/config.json"
10
11 // I2C
12 #define I2C_ADDRESS 0x50
13 #define I2C_DEVICE "/dev/i2c-1"
14
15 // MODBUS
16 #define SIWAREX_ADDRESS 0x14
17
18 // LED Pins
19 #define PIN_LED_READY 23
20 #define PIN_LED_BUSY 24
21 #define PIN_LED_FAULT 25
22
23 // PWM Pin
24 #define PWM_PIN 13
25
26 // 24V Power Switch Pins
27 #define PIN_POWERSW1 4
28 #define PIN_POWERSW2 26
29
30 // Pins for Impedance switching
31 #define PIN_IMPEDANCE1 5
32 #define PIN_IMPEDANCE2 6
33
34 // Pins for extra switches (e.g. WebServer, WriteProtect)
35 #define PIN_EXTRASW1 27
36 #define PIN_EXTRASW2 22
37
38 // Analog Channels
39 #define ADDVOL_CHANNEL 2
40 #define SUBVOL_CHANNEL 3
41 #define CELL_DC 1
42 #define SEN_OUT 4
43 #define EXC_IN 1
44 #define SEN_IN 2
45
46 enum LoadCellMode
47 {
48     NORMAL = 0x00,
49     OVERLOAD = 0x01,
50     INVERTED = 0x02,
51 };
52 typedef LoadCellMode;
53
54 enum IMPEDANCE
55 {
56     OPEN = 0x00,
57     NOMINAL = 0x01,
58     SHORT = 0x02,
59 };
60 typedef IMPEDANCE;
61
62 enum SYSTEM_TYPE
63 {
64     DOSING_SCALE = 0x01,
65     BELT_SCALE = 0x02,
66 };
67 typedef SYSTEM_TYPE;
68
69 enum MATERIAL_FLOW
70 {
71     NONE = 0x00,
72     EMPTY,
73     FINE,
74     COARSE,
75     XCOARSE,
76 };
77 typedef MATERIAL_FLOW;
78
79 enum RUN_MODE
80 {
81     AUTO,
82     PASSIVE,
83     MANUAL,
84     IDLE,
85     OFF
86 };
87 typedef RUN_MODE;
88
89 struct CUBIC_FUNCTION
90 {
91     float a, b, c, d;

```

```

109 } typedef CUBIC_FUNCTION;
110
111 class Configuration
112 {
113 public:
114     Configuration(std::string path);
115     ~Configuration();
116
117     void loadConfiguration();
118
119     // SETTING VARIABLES
120     LoadCellMode cellMode = LoadCellMode::NORMAL;
121     SYSTEM_TYPE systemType = SYSTEM_TYPE::DOSING_SCALE;
122     float exc_voltage = 10.f;
123     float load_weight = 20.f;
124     float initial_weight = 10.f;
125     float addvol_ratio = 500;
126     float max_diff_voltage = 40;
127     float cellCharecteristic = 4;
128     float speedAt100 = 5;
129     float freqAt100 = 10000;
130
131     float startVoltage = 2;
132     float endVoltage = 9;
133
134     float a, b, c, d;
135     CUBIC_FUNCTION calibrationReg;
136
137     // Input channel assignment
138     MATERIAL_FLOW inputChannel1 = MATERIAL_FLOW::EMPTY;
139     MATERIAL_FLOW inputChannel2 = MATERIAL_FLOW::FINE;
140     MATERIAL_FLOW inputChannel3 = MATERIAL_FLOW::COARSE;
141     MATERIAL_FLOW inputChannel4 = MATERIAL_FLOW::XCOARSE;
142
143 private:
144     void parseJSON();
145     std::string _path;
146 };

```

4.4 F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.cpp File Reference

```
#include "GPIO.hpp"
```

4.5 F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.hpp File Reference

```

#include <signal.h>
#include <pigpio.h>
#include <stdint.h>
#include <cstdio>

```

Classes

- class [GPIO](#)

4.6 GPIO.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #include <signal.h>
3 #include <pigpio.h>
4 #include <stdint.h>

```

```

5 #include <cstdio>
6
7 class GPIO
8 {
9 public:
10     GPIO();
11     ~GPIO();
12     void setPWM(int pin, float dutyCycle, float frequency);
13
14     void setPinMode(uint8_t pin, uint8_t mode);
15
16     void writePin(uint8_t pin, bool state);
17     bool readPin(uint8_t pin);
18
19 private:
20 };

```

4.7 F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.cpp File Reference

```
#include "I2C.hpp"
```

4.8 F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.hpp File Reference

```

#include <stdio.h>
#include <unistd.h>
#include <string>
#include <stdint.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>

```

Classes

- class [I2C](#)

4.9 I2C.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #include <stdio.h>
3 #include <unistd.h>
4 #include <string>
5 #include <stdint.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
8 #include <sys/ioctl.h>
9 #include <linux/i2c.h>
10 #include <linux/i2c-dev.h>
11
12 class I2C
13 {
14 public:
15     I2C(std::string dev, uint16_t address);
16     ~I2C();
17     bool begin();
18     bool writeData(uint8_t data);

```

```

19     bool writeData(uint8_t *data, uint8_t length);
20     bool readData(uint8_t *data, uint8_t length);
21     uint8_t readData();
22
23 private:
24     std::string _dev;
25     uint16_t _address;
26     int i2c0 = -1;
27 };

```

4.10 F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.cpp File Reference

```
#include "IABoard.hpp"
```

4.11 F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.hpp File Reference

```

#include "Configuration.hpp"
#include "I2C.hpp"
#include "utility.hpp"
#include <chrono>
#include <thread>
#include <iostream>

```

Classes

- class [IABoard](#)

Enumerations

- enum [TRANSITION](#) {
[DISABLE](#) = 0x00 , [RISING](#) = 0x01 , [FALLING](#) = 0x02 , [BOTH](#) = 0x03 ,
[UNDEFINED](#) = 0x04 }

4.11.1 Enumeration Type Documentation

4.11.1.1 TRANSITION

```
enum TRANSITION
```

Enumerator

DISABLE	
RISING	
FALLING	
BOTH	
UNDEFINED	

4.12 IABoard.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #include "Configuration.hpp"
3 #include "I2C.hpp"
4 #include "utility.hpp"
5
6 #include <chrono>
7 #include <thread>
8 #include <iostream>
9 using namespace std::chrono_literals;
10
11 enum TRANSITION
12 {
13     DISABLE = 0x00,
14     RISING = 0x01,
15     FALLING = 0x02,
16     BOTH = 0x03,
17     UNDEFINED = 0x04
18 } typedef TRANSITION;
19
20 class IABoard
21 {
22 public:
23     IABoard();
24     ~IABoard();
25
26     // Check if the board is responding
27     bool detectBoard();
28
29     // Read all digital inputs
30     uint8_t digitalRead();
31     // Read digital input of certain channel 1 - 4
32     bool digitalRead(uint8_t channel);
33     bool getDigitalRead(uint8_t channel);
34
35     // Reads the number of counted transitions (if enabled)
36     uint16_t readTransistions(uint8_t channel);
37     // Reads the currently set transition type
38     TRANSITION getTransistionType(uint8_t channel);
39     // Sets the type of transistions that should be counted
40     void setTransistionType(uint8_t channel, TRANSITION tran);
41     // Sets the transistion counter of a channel to 0
42     void resetTransitions(uint8_t channel);
43
44     // Get the currently set analog output voltage
45     float getAnalogVolOut(uint8_t channel);
46     // Set the analog output voltage from 0 - 10V, voltage in volts
47     void setAnalogVolOut(uint8_t channel, float voltage);
48
49     // Get the currently set analog output current
50     float getAnalogCurOut(uint8_t channel);
51     // Set the analog output current from 4 - 20mA, current in mA
52     void setAnalogCurOut(uint8_t channel, float current);
53
54     // Get the PWM Duty Cycle for the Open Drain Output (if not used as digital out)
55     float getOpenDrainPWM(uint8_t channel);
56     // Set the PWM Duty Cycle (0 - 100%) for the Open Drain Output
57     void setOpenDrainPWM(uint8_t channel, float dutyCycle);
58
59     // Read all digital open drain outputs
60     uint8_t getOpenDrainDOUT();
61     // Get the currently set open drain digital out value
62     bool getOpenDrainDOUT(uint8_t channel);
63     // Set the digital open drain output
64     void setOpenDrainDOUT(uint8_t channel, bool value);
65
66     // Gets the state of a certain LED
67     bool getLED(uint8_t channel);
68     // Sets a certain LED Low or High
69     void setLED(uint8_t channel, bool value);
70     // Sets all LEDs ON or OFF
71     void setAllLED(bool value);
72
73     // Reads the analog input voltage of a certain channel (0-10V)
74     float readAnalogVolIn(uint8_t channel);
75     // Reads the analog input voltage of a certain channel (-10-10V, Jumper set)
76     float readAnalogVolInPM(uint8_t channel);
77
78     // Reads the analog input current of a certain channel (4-20mA)
79     float readAnalogCurIn(uint8_t channel);
80
81     void getBoardData();
82     void getBoardData(uint8_t *temp, float *rail24, float *rail5);

```

```

83
84 // Turn all digital and analog outputs off
85 void setAllOff();
86
87 private:
88     I2C *_i2c;
89
90     bool _digitalRead[4] = {0, 0, 0, 0};
91
92     uint8_t _fwVersion[2] = {0x00, 0x00};
93     uint8_t _boardTemperature = 0;
94     float _24Vrail = 0.f;
95     float _5Vrail = 0.f;
96
97 // Delay because the IA-Board can only handle commands every few ms
98 const std::chrono::milliseconds _delayBetweenCommands = 2ms;
99 std::chrono::time_point<std::chrono::system_clock, std::chrono::duration<double> _lastCommand;
100
101 // Wait till the minimum time between commands has elapsed
102 void waitForIA();
103 };

```

4.13 F:/GITHUB/SiWaSIM-PiSoftware/src/main.cpp File Reference

```

#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include "I2C.hpp"
#include "UART.hpp"
#include "GPIO.hpp"
#include "IABoard.hpp"
#include "PCB.hpp"
#include "Simulator.hpp"
#include "Modbus.hpp"
#include "SIWAREX.hpp"
#include "matplotlib/matplotlibcpp.h"

```

Functions

- int [main](#) ()

4.13.1 Function Documentation

4.13.1.1 main()

```
int main ( )
```

4.14 F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.cpp File Reference

```
#include "MaterialFlow.hpp"
```


4.15 F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.hpp File Reference

```
#include <iostream>
#include "Configuration.hpp"
#include "IABoard.hpp"
```

Classes

- struct [CURVE](#)
- class [MaterialFlow](#)

4.16 MaterialFlow.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #include <iostream>
3 #include "Configuration.hpp"
4 #include "IABoard.hpp"
5
6 struct CURVE
7 {
8     float startDelay = 0;
9     float stopDelay = 0;
10    float riseTime = 1;
11    float fallTime = 1;
12    float maxFlow = 1;
13 } typedef CURVE;
14
15 class MaterialFlow
16 {
17 public:
18     MaterialFlow(uint8_t channel);
19     MaterialFlow(uint8_t channel, MATERIAL_FLOW flowType);
20     ~MaterialFlow();
21
22     void setFlowCurve(CURVE curve);
23     void setFlowType(MATERIAL_FLOW flowType);
24     float update(float *currentWeight, float dt, bool pinState);
25
26 private:
27     uint8_t _channel;
28     MATERIAL_FLOW _flowType;
29
30     bool _lastPinState = 0;
31     float _currentFlow = 0;
32     float _lastPinStateTime = 0;
33
34     CURVE _curve;
35
36     IABoard *_ia;
37 };
```

4.17 F:/GITHUB/SiWaSIM-PiSoftware/src/Modbus.cpp File Reference

```
#include "Modbus.hpp"
```

4.18 F:/GITHUB/SiWaSIM-PiSoftware/src/Modbus.hpp File Reference

```
#include <vector>
#include <iostream>
#include "Configuration.hpp"
#include "UART.hpp"
```

Classes

- class [Modbus](#)

4.19 Modbus.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #include <vector>
3 #include <iostream>
4 #include "Configuration.hpp"
5
6 #include "UART.hpp"
7
8 class Modbus
9 {
10 public:
11     Modbus();
12     ~Modbus();
13
14     void transmitRequest(uint16_t startRegister, uint16_t length);
15     std::vector<uint8_t> receiveResponse();
16
17 private:
18     uint16_t calculateCRC(uint8_t *data, int length);
19
20     UART *_uart;
21     const uint8_t _address = SIWAREX_ADDRESS;
22 };
```

4.20 F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.cpp File Reference

```
#include "PCB.hpp"
```

4.21 F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.hpp File Reference

```
#include "utility.hpp"
#include "GPIO.hpp"
#include "IABoard.hpp"
#include "Configuration.hpp"
```

Classes

- class [PCB](#)

4.22 PCB.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #include "utility.hpp"
3 #include "GPIO.hpp"
4 #include "IABoard.hpp"
5 #include "Configuration.hpp"
6
7 class PCB
8 {
9 public:
10     PCB(Configuration *config);
11     ~PCB();
12
13     void ledFault(bool state);
14     void ledBusy(bool state);
15     void ledReady(bool state);
16
17     void setImpedance(IMPEDANCE impedance);
18
19     void setEXTRASW1(bool state);
20     void setEXTRASW2(bool state);
21
22     void setPOWERSW1(bool state);
23     void setPOWERSW2(bool state);
24
25     void setLoadcellVoltage(float voltage);
26     void setLoadcellDCVoltage(float voltage);
27     void setCellAddvol(float voltage);
28     void setCellSubvol(float voltage);
29
30     void setSENVoltage(float voltage);
31
32     float getEXCVoltage();
33     float getSENVoltage();
34
35     void setPWM(float frequency, float dutyCycle);
36     void getBoardStatus();
37
38     void reloadConfig();
39
40     void setAllof();
41
42 private:
43     GPIO *_gpio;
44     IABoard *_ia;
45     Configuration *_config;
46 };

```

4.23 F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.cpp File Reference

```
#include "Simulator.hpp"
```

4.24 F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.hpp File Reference

```

#include "PCB.hpp"
#include "Configuration.hpp"
#include "IABoard.hpp"
#include "MaterialFlow.hpp"
#include "SIWAREX.hpp"
#include <chrono>
#include <thread>

```

Classes

- class [Simulator](#)

4.25 Simulator.hpp

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #include "PCB.hpp"
3 #include "Configuration.hpp"
4 #include "IABoard.hpp"
5 #include "MaterialFlow.hpp"
6 #include "SIWAREX.hpp"
7
8 #include <chrono>
9 #include <thread>
10
11 using namespace std::chrono_literals;
12
13 class Simulator
14 {
15 public:
16     Simulator();
17     ~Simulator();
18
19     void setWeightPER(float percentage); // Set the weight from 0 - 100% of nominal Load
20     void setWeightKG(float kg);         // Set the weight in kg
21
22     void setVelocity(float meterspersecond);
23     void setVelocityPER(float percentage);
24     void setVelocityFRQ(float frequency);
25
26     void setImpedance(IMPEDANCE impedance);
27
28     void bootupAnimation();
29     void reloadConfig();
30
31     void testFunction();
32
33     float run(RUN_MODE runMode, float timestep, float *weight);
34     float runPassive(float timestep, float *weight);
35
36     void calibrateLCVoltage(bool autoCalib);
37
38 private:
39     Configuration *_config;
40     PCB *_pcb;
41     IABoard *_ia;
42     SIWAREX *_siwarex;
43
44     MaterialFlow *_materialFlows[4];
45 };

```

4.26 F:/GITHUB/SiWaSIM-PiSoftware/src/SIWAREX.cpp File Reference

```
#include "SIWAREX.hpp"
```

4.27 F:/GITHUB/SiWaSIM-PiSoftware/src/SIWAREX.hpp File Reference

```

#include <vector>
#include "SIWAREX_REGISTER.hpp"
#include "Modbus.hpp"
#include "utility.hpp"

```

Classes

- class [SIWAREX](#)

4.28 SIWAREX.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #include <vector>
3
4 #include "SIWAREX_REGISTER.hpp"
5 #include "Modbus.hpp"
6 #include "utility.hpp"
7
8 class SIWAREX
9 {
10 public:
11     SIWAREX();
12     ~SIWAREX();
13
14     float getLoadcellVoltage();
15
16 private:
17     float requestFloat(uint16_t startRegister);
18
19     Modbus *_modbus;
20 };
```

4.29 F:/GITHUB/SiWaSIM-PiSoftware/src/SIWAREX_REGISTER.hpp File Reference

Macros

- #define [LOADCELL_VOLTAGE](#) 3058

4.29.1 Macro Definition Documentation

4.29.1.1 LOADCELL_VOLTAGE

```
#define LOADCELL_VOLTAGE 3058
```

4.30 SIWAREX_REGISTER.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #define LOADCELL_VOLTAGE 3058
```

4.31 F:/GITHUB/SiWaSIM-PiSoftware/src/UART.cpp File Reference

```
#include "UART.hpp"
```

4.32 F:/GITHUB/SiWaSIM-PiSoftware/src/UART.hpp File Reference

```
#include <stdint.h>
#include <fcntl.h>
#include <iostream>
#include <sstream>
#include <termios.h>
#include <unistd.h>
#include <vector>
```

Classes

- class [UART](#)

4.33 UART.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #include <stdint.h>
3 #include <fcntl.h>
4 #include <iostream>
5 #include <sstream>
6 #include <termios.h>
7 #include <unistd.h>
8 #include <vector>
9
10 class UART
11 {
12 public:
13     UART();
14     ~UART();
15     bool begin();
16     bool transmitMSG(uint8_t *msg, uint16_t length);
17     std::vector<uint8_t> receiveMSG();
18
19 private:
20     int uart0 = -1;
21     // std::string _dev;
22     const uint8_t _messageSizeRX = 0; // Number of bytes to wait for
23     const uint8_t _messageTimeoutRX = 1; // Read Timeout in 0.1s steps
24 };
```

4.34 F:/GITHUB/SiWaSIM-PiSoftware/src/utility.cpp File Reference

```
#include "utility.hpp"
```

Functions

- float [constrainMinMax](#) (float value, float min, float max)
- float [constrainMin](#) (float value, float min)
- float [constrainMax](#) (float value, float max)
- void [linearRegression](#) (std::vector< float > x, std::vector< float > y, float *m, float *b)
- float [calculateAverage](#) (std::vector< float > values)
- void [cubicRegression](#) (std::vector< float > x, std::vector< float > y, float *a, float *b, float *c, float *d)
- float [solveCubicForVoltage](#) (float a, float b, float c, float d, float value)
- float [calculateCubic](#) (float a, float b, float c, float d, float x)
- float [calculateCubicDeriv](#) (float a, float b, float c, float x)
- float [bytesToFloat](#) (uint8_t *bytes)
- float [bytesToFloat](#) (uint8_t b3, uint8_t b2, uint8_t b1, uint8_t b0)
- void [delay](#) (std::chrono::milliseconds delayMS)

4.34.1 Function Documentation

4.34.1.1 bytesToFloat() [1/2]

```
float bytesToFloat (
    uint8_t * bytes )
```

4.34.1.2 bytesToFloat() [2/2]

```
float bytesToFloat (
    uint8_t b3,
    uint8_t b2,
    uint8_t b1,
    uint8_t b0 )
```

4.34.1.3 calculateAverage()

```
float calculateAverage (
    std::vector< float > values )
```

Calculates the average of values of a vector

Parameters

<i>values</i>	The vector that contains the values
---------------	-------------------------------------

Returns

Returns the average of the values in the vector

4.34.1.4 calculateCubic()

```
float calculateCubic (
    float a,
    float b,
    float c,
    float d,
    float x )
```

Calculates a y-value of a cubic function with $f(x) = ax^3 + bx^2 + cx + d$

Parameters

<i>a</i>	Coefficient in front of x^3
<i>b</i>	Coefficient in front of x^2
<i>c</i>	Coefficient in front of x^1
<i>d</i>	Coefficient in front of x^0
<i>x</i>	X-value for which the function should be calculated

Returns

Returns the y-value $y = f(x)$

4.34.1.5 calculateCubicDeriv()

```
float calculateCubicDeriv (
    float a,
    float b,
    float c,
    float x )
```

Calculates a y-value of a cubic derivative function $f'(x)$ with $f(x) = ax^3 + bx^2 + cx + d$ and $f'(x) = 3ax^2 + 2bx + c$

Parameters

<i>a</i>	Coefficient in front of x^3
<i>b</i>	Coefficient in front of x^2
<i>c</i>	Coefficient in front of x^1
<i>x</i>	X-value for which the function should be calculated

Returns

Returns the y-value $y = f'(x)$

4.34.1.6 constrainMax()

```
float constrainMax (
    float value,
    float max )
```

Constrain a value to an upper limit if the value is above that limit

Parameters

<i>value</i>	The value to be clipped
<i>max</i>	The upper limit

Returns

Returns the clipped / constrained value

4.34.1.7 constrainMin()

```
float constrainMin (
    float value,
    float min )
```

Constrain a value to a lower limit if the value is below that limit

Parameters

<i>value</i>	The value to be clipped
<i>min</i>	The lower limit

Returns

Returns the clipped / constrained value

4.34.1.8 constrainMinMax()

```
float constrainMinMax (
    float value,
    float min,
    float max )
```

Constrain a value between an upper and a lower limit to clip the value

Parameters

<i>value</i>	The value to be clipped
<i>min</i>	The lower limit
<i>max</i>	The upper limit

Returns

Returns the clipped / constrained value

4.34.1.9 cubicRegression()

```
void cubicRegression (
    std::vector< float > x,
```

```
std::vector< float > y,
float * a,
float * b,
float * c,
float * d )
```

Calculates a cubic regression $f(x)=ax^3+bx^2+cx+d$ for a dataset of x and y values

Parameters

<i>x</i>	Vector of x-values of the dataset
<i>y</i>	Vector of y-values of the dataset
<i>a</i>	Coefficient in front of x^3
<i>b</i>	Coefficient in front of x^2
<i>c</i>	Coefficient in front of x^1
<i>d</i>	Coefficient in front of x^0

Calculates a cubic regression based on the following instructions: <https://www.omnicalculator.com/statistics/cubic-regression>

The formula for the result vector which contains the function coefficients a, b, c, d is

$$result = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

where...

X is the base matrix X^T is the transposed base matrix \cdot represents the matrix multiplication $^{-1}$ represents the inverse of a matrix y is the vector that contains all y-values in the same order as the x-values in the base matrix

4.34.1.10 delay()

```
void delay (
    std::chrono::milliseconds delayMS )
```

4.34.1.11 linearRegression()

```
void linearRegression (
    std::vector< float > x,
    std::vector< float > y,
    float * m,
    float * b )
```

Calculates a linear regression $f(x) = mx + b$ for a dataset of x and y values

Parameters

<i>x</i>	Vector of x-values of the dataset
<i>y</i>	Vector of y-values of the dataset
<i>m</i>	Pointer to the slope m
<i>b</i>	Pointer to the y-intercept b

4.34.1.12 solveCubicForVoltage()

```
float solveCubicForVoltage (
    float a,
    float b,
    float c,
    float d,
    float value )
```

Newton-Raphson Method for finding the x-value that corresponds to a y-value of a cubic function $f(x) = ax^3 + x^2 + cx + d = value$, only for the range 0 - 10

Parameters

<i>a</i>	Coefficient in front of x^3
<i>b</i>	Coefficient in front of x^2
<i>c</i>	Coefficient in front of x^1
<i>d</i>	Coefficient in front of x^0
<i>value</i>	y-value of the cubic function that corresponds to the wanted x-value

Returns

Returns the x-value that corresponds to the y-value

4.35 F:/GITHUB/SiWaSIM-PiSoftware/src/utility.hpp File Reference

```
#include <vector>
#include <iostream>
#include <cmath>
#include <chrono>
#include <thread>
#include "Eigen/Dense"
```

Functions

- float [constrainMinMax](#) (float value, float min, float max)
- float [constrainMin](#) (float value, float min)
- float [constrainMax](#) (float value, float max)
- void [linearRegression](#) (std::vector< float > x, std::vector< float > y, float *a, float *b)
- float [calculateAverage](#) (std::vector< float > values)
- void [cubicRegression](#) (std::vector< float > x, std::vector< float > y, float *a, float *b, float *c, float *d)
- float [solveCubicForVoltage](#) (float a, float b, float c, float d, float value)
- float [calculateCubic](#) (float a, float b, float c, float d, float x)
- float [calculateCubicDeriv](#) (float a, float b, float c, float x)
- float [bytesToFloat](#) (uint8_t *bytes)
- float [bytesToFloat](#) (uint8_t b3, uint8_t b2, uint8_t b1, uint8_t b0)
- void [delay](#) (std::chrono::milliseconds delayMS)

4.35.1 Function Documentation

4.35.1.1 bytesToFloat() [1/2]

```
float bytesToFloat (
    uint8_t * bytes )
```

4.35.1.2 bytesToFloat() [2/2]

```
float bytesToFloat (
    uint8_t b3,
    uint8_t b2,
    uint8_t b1,
    uint8_t b0 )
```

4.35.1.3 calculateAverage()

```
float calculateAverage (
    std::vector< float > values )
```

Calculates the average of values of a vector

Parameters

<i>values</i>	The vector that contains the values
---------------	-------------------------------------

Returns

Returns the average of the values in the vector

4.35.1.4 calculateCubic()

```
float calculateCubic (
    float a,
    float b,
    float c,
    float d,
    float x )
```

Calculates a y-value of a cubic function with $f(x) = ax^3 + bx^2 + cx + d$

Parameters

<i>a</i>	Coefficient in front of x^3
<i>b</i>	Coefficient in front of x^2
<i>c</i>	Coefficient in front of x^1
<i>d</i>	Coefficient in front of x^0
<i>x</i>	X-value for which the function should be calculated

Returns

Returns the y-value $y = f(x)$

4.35.1.5 calculateCubicDeriv()

```
float calculateCubicDeriv (
    float a,
    float b,
    float c,
    float x )
```

Calculates a y-value of a cubic derivative function $f'(x)$ with $f(x) = ax^3 + bx^2 + cx + d$ and $f'(x) = 3ax^2 + 2bx + c$

Parameters

<i>a</i>	Coefficient in front of x^3
<i>b</i>	Coefficient in front of x^2
<i>c</i>	Coefficient in front of x^1
<i>x</i>	X-value for which the function should be calculated

Returns

Returns the y-value $y = f'(x)$

4.35.1.6 constrainMax()

```
float constrainMax (
    float value,
    float max )
```

Constrain a value to an upper limit if the value is above that limit

Parameters

<i>value</i>	The value to be clipped
<i>max</i>	The upper limit

Returns

Returns the clipped / constrained value

4.35.1.7 constrainMin()

```
float constrainMin (
    float value,
    float min )
```

Constrain a value to a lower limit if the value is below that limit

Parameters

<i>value</i>	The value to be clipped
<i>min</i>	The lower limit

Returns

Returns the clipped / constrained value

4.35.1.8 constrainMinMax()

```
float constrainMinMax (
    float value,
    float min,
    float max )
```

Constrain a value between an upper and a lower limit to clip the value

Parameters

<i>value</i>	The value to be clipped
<i>min</i>	The lower limit
<i>max</i>	The upper limit

Returns

Returns the clipped / constrained value

4.35.1.9 cubicRegression()

```
void cubicRegression (
    std::vector< float > x,
```

```
std::vector< float > y,
float * a,
float * b,
float * c,
float * d )
```

Calculates a cubic regression $f(x)=ax^3+bx^2+cx+d$ for a dataset of x and y values

Parameters

<i>x</i>	Vector of x-values of the dataset
<i>y</i>	Vector of y-values of the dataset
<i>a</i>	Coefficient in front of x^3
<i>b</i>	Coefficient in front of x^2
<i>c</i>	Coefficient in front of x^1
<i>d</i>	Coefficient in front of x^0

Calculates a cubic regression based on the following instructions: <https://www.omnicalculator.com/statistics/cubic-regression>

The formula for the result vector which contains the function coefficients a, b, c, d is

$$result = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

where...

X is the base matrix X^T is the transposed base matrix \cdot represents the matrix multiplication $^{-1}$ represents the inverse of a matrix y is the vector that contains all y-values in the same order as the x-values in the base matrix

4.35.1.10 delay()

```
void delay (
    std::chrono::milliseconds delayMS )
```

4.35.1.11 linearRegression()

```
void linearRegression (
    std::vector< float > x,
    std::vector< float > y,
    float * m,
    float * b )
```

Calculates a linear regression $f(x) = mx + b$ for a dataset of x and y values

Parameters

<i>x</i>	Vector of x-values of the dataset
<i>y</i>	Vector of y-values of the dataset
<i>m</i>	Pointer to the slope m
<i>b</i>	Pointer to the y-intercept b

4.35.1.12 solveCubicForVoltage()

```
float solveCubicForVoltage (
    float a,
    float b,
    float c,
    float d,
    float value )
```

Newton-Raphson Method for finding the x-value that corresponds to a y-value of a cubic function $f(x) = ax^3 + x^2 + cx + d = value$, only for the range 0 - 10

Parameters

<i>a</i>	Coefficient in front of x^3
<i>b</i>	Coefficient in front of x^2
<i>c</i>	Coefficient in front of x^1
<i>d</i>	Coefficient in front of x^0
<i>value</i>	y-value of the cubic function that corresponds to the wanted x-value

Returns

Returns the x-value that corresponds to the y-value

4.36 utility.hpp

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #include <vector>
3 #include <iostream>
4 #include <cmath>
5 #include <chrono>
6 #include <thread>
7
8 using namespace std::chrono_literals;
9
10 #include "Eigen/Dense"
11
12 using Eigen::MatrixXd;
13 using Eigen::VectorXd;
14
15 float constrainMinMax(float value, float min, float max);
16 float constrainMin(float value, float min);
17 float constrainMax(float value, float max);
18
19 void linearRegression(std::vector<float> x, std::vector<float> y, float *m, float *b);
20
21 float calculateAverage(std::vector<float> values);
22
23 void cubicRegression(std::vector<float> x, std::vector<float> y, float *a, float *b, float *c, float *d);
24
25 float solveCubicForVoltage(float a, float b, float c, float d, float value);
26
27 float calculateCubic(float a, float b, float c, float d, float x);
28 float calculateCubicDeriv(float a, float b, float c, float x);
29
30 float bytesToFloat(uint8_t *bytes);
31
32 float bytesToFloat(uint8_t b3, uint8_t b2, uint8_t b1, uint8_t b0);
33
34 void delay(std::chrono::milliseconds delayMS);
```


Index

- ~Configuration
 - Configuration, [5](#)
- ~GPIO
 - GPIO, [7](#)
- ~I2C
 - I2C, [9](#)
- ~IABoard
 - IABoard, [10](#)
- ~PCB
 - PCB, [15](#)
- ~Simulator
 - Simulator, [17](#)
- ~UART
 - UART, [18](#)
- ADDVOL_CHANNEL
 - PCB.hpp, [28](#)
- addvol_ratio
 - Configuration, [6](#)
- begin
 - I2C, [9](#)
 - UART, [18](#)
- BOTH
 - IABoard.hpp, [25](#)
- cellCharecteristic
 - Configuration, [6](#)
- cellMode
 - Configuration, [6](#)
- Configuration, [5](#)
 - ~Configuration, [5](#)
 - addvol_ratio, [6](#)
 - cellCharecteristic, [6](#)
 - cellMode, [6](#)
 - Configuration, [5](#)
 - exc_voltage, [6](#)
 - initial_weight, [6](#)
 - load_weight, [6](#)
 - loadConfiguration, [6](#)
 - max_diff_voltage, [7](#)
- Configuration.hpp
 - IMPEDANCE, [21](#)
 - INVERTED, [22](#)
 - LoadCellMode, [22](#)
 - NOMINAL, [22](#)
 - NORMAL, [22](#)
 - OPEN, [22](#)
 - OVERLOAD, [22](#)
 - SHORT, [22](#)
- constrainMax
 - utility.cpp, [31](#)
 - utility.hpp, [32](#)
- constrainMin
 - utility.cpp, [32](#)
 - utility.hpp, [32](#)
- constrainMinMax
 - utility.cpp, [32](#)
 - utility.hpp, [33](#)
- detectBoard
 - IABoard, [11](#)
- digitalRead
 - IABoard, [11](#)
- DISABLE
 - IABoard.hpp, [25](#)
- exc_voltage
 - Configuration, [6](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.cpp, [21](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.hpp, [21](#), [22](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.cpp, [23](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.hpp, [23](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.cpp, [23](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.hpp, [23](#), [24](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.cpp, [24](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.hpp, [24](#), [25](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/main.cpp, [26](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.cpp, [27](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.hpp, [27](#), [29](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.cpp, [30](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.hpp, [30](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/UART.cpp, [30](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/UART.hpp, [31](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/utility.cpp, [31](#)
- F:/GITHUB/SiWaSIM-PiSoftware/src/utility.hpp, [32](#), [33](#)
- FALLING
 - IABoard.hpp, [25](#)
- getAnalogCurOut
 - IABoard, [11](#)
- getAnalogVolOut
 - IABoard, [11](#)
- getEXCVoltage
 - PCB, [15](#)
- getLED

- IABoard, 11
- getOpenDrainDOUT
 - IABoard, 11, 12
- getOpenDrainPWM
 - IABoard, 12
- getSENVoltage
 - PCB, 15
- getTransistionType
 - IABoard, 12
- GPIO, 7
 - ~GPIO, 7
 - GPIO, 7
 - readPin, 7
 - setPinMode, 8
 - setPWM, 8
 - writePin, 8
- I2C, 8
 - ~I2C, 9
 - begin, 9
 - I2C, 9
 - readData, 9
 - writeData, 9
- I2C_ADDRESS
 - IABoard.hpp, 25
- IABoard, 10
 - ~IABoard, 10
 - detectBoard, 11
 - digitalRead, 11
 - getAnalogCurOut, 11
 - getAnalogVolOut, 11
 - getLED, 11
 - getOpenDrainDOUT, 11, 12
 - getOpenDrainPWM, 12
 - getTransistionType, 12
 - IABoard, 10
 - readAnalogCurIn, 12
 - readAnalogVolIn, 12
 - readAnalogVolInPM, 12
 - readTransistions, 12
 - resetTransitions, 13
 - setAnalogCurOut, 13
 - setAnalogVolOut, 13
 - setLED, 13
 - setOpenDrainDOUT, 13
 - setOpenDrainPWM, 13
 - setTransistionType, 14
- IABoard.hpp
 - BOTH, 25
 - DISABLE, 25
 - FALLING, 25
 - I2C_ADDRESS, 25
 - RISING, 25
 - TRANSITION, 25
 - UNDEFINED, 25
- IMPEDANCE
 - Configuration.hpp, 21
- initial_weight
 - Configuration, 6
- INVERTED
 - Configuration.hpp, 22
- ledBusy
 - PCB, 15
- ledFault
 - PCB, 15
- ledReady
 - PCB, 15
- load_weight
 - Configuration, 6
- LoadCellMode
 - Configuration.hpp, 22
- loadConfiguration
 - Configuration, 6
- main
 - main.cpp, 27
- main.cpp
 - main, 27
- max_diff_voltage
 - Configuration, 7
- NOMINAL
 - Configuration.hpp, 22
- NORMAL
 - Configuration.hpp, 22
- OPEN
 - Configuration.hpp, 22
- OVERLOAD
 - Configuration.hpp, 22
- PCB, 14
 - ~PCB, 15
 - getEXCVoltage, 15
 - getSENVoltage, 15
 - ledBusy, 15
 - ledFault, 15
 - ledReady, 15
 - PCB, 14
 - setEXTRASW1, 15
 - setEXTRASW2, 16
 - setImpedance, 16
 - setLoadcellIDCVoltage, 16
 - setLoadcellVoltage, 16
 - setPOWERSW1, 16
 - setPOWERSW2, 16
 - setSENVoltage, 16
- PCB.hpp
 - ADDVOL_CHANNEL, 28
 - PIN_EXTRASW1, 28
 - PIN_EXTRASW2, 28
 - PIN_IMPEDANCE1, 28
 - PIN_IMPEDANCE2, 28
 - PIN_LED_BUSY, 28
 - PIN_LED_FAULT, 28
 - PIN_LED_READY, 28
 - PIN_POWERSW1, 29

- PIN_POWERSW2, [29](#)
- SUBVOL_CHANNEL, [29](#)
- PIN_EXTRASW1
 - PCB.hpp, [28](#)
- PIN_EXTRASW2
 - PCB.hpp, [28](#)
- PIN_IMPEDANCE1
 - PCB.hpp, [28](#)
- PIN_IMPEDANCE2
 - PCB.hpp, [28](#)
- PIN_LED_BUSY
 - PCB.hpp, [28](#)
- PIN_LED_FAULT
 - PCB.hpp, [28](#)
- PIN_LED_READY
 - PCB.hpp, [28](#)
- PIN_POWERSW1
 - PCB.hpp, [29](#)
- PIN_POWERSW2
 - PCB.hpp, [29](#)
- readAnalogCurln
 - IABoard, [12](#)
- readAnalogVolln
 - IABoard, [12](#)
- readAnalogVollnPM
 - IABoard, [12](#)
- readData
 - I2C, [9](#)
- readPin
 - GPIO, [7](#)
- readTransistions
 - IABoard, [12](#)
- receiveMSG
 - UART, [18](#)
- resetTransitions
 - IABoard, [13](#)
- RISING
 - IABoard.hpp, [25](#)
- setAnalogCurOut
 - IABoard, [13](#)
- setAnalogVolOut
 - IABoard, [13](#)
- setEXTRASW1
 - PCB, [15](#)
- setEXTRASW2
 - PCB, [16](#)
- setImpedance
 - PCB, [16](#)
- setLED
 - IABoard, [13](#)
- setLoadcellDCVoltage
 - PCB, [16](#)
- setLoadcellVoltage
 - PCB, [16](#)
- setOpenDrainDOUT
 - IABoard, [13](#)
- setOpenDrainPWM
 - IABoard, [13](#)
- setPinMode
 - GPIO, [8](#)
- setPOWERSW1
 - PCB, [16](#)
- setPOWERSW2
 - PCB, [16](#)
- setPWM
 - GPIO, [8](#)
- setSENVoltage
 - PCB, [16](#)
- setTransistionType
 - IABoard, [14](#)
- setWeightKG
 - Simulator, [17](#)
- setWeightPER
 - Simulator, [17](#)
- SHORT
 - Configuration.hpp, [22](#)
- Simulator, [17](#)
 - ~Simulator, [17](#)
 - setWeightKG, [17](#)
 - setWeightPER, [17](#)
 - Simulator, [17](#)
- SUBVOL_CHANNEL
 - PCB.hpp, [29](#)
- TRANSITION
 - IABoard.hpp, [25](#)
- transmitMSG
 - UART, [19](#)
- UART, [18](#)
 - ~UART, [18](#)
 - begin, [18](#)
 - receiveMSG, [18](#)
 - transmitMSG, [19](#)
 - UART, [18](#)
- UNDEFINED
 - IABoard.hpp, [25](#)
- utility.cpp
 - constrainMax, [31](#)
 - constrainMin, [32](#)
 - constrainMinMax, [32](#)
- utility.hpp
 - constrainMax, [32](#)
 - constrainMin, [32](#)
 - constrainMinMax, [33](#)
- writeData
 - I2C, [9](#)
- writePin
 - GPIO, [8](#)