SiWaSim

Generated by Doxygen 1.9.3

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

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

path	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 readAnalogVolIn (uint8_t channel)
- float readAnalogVolInPM (uint8_t channel)
- float readAnalogCurIn (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

channel	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

channel	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

channel	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

channel	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

value	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

channel	The LED to be toggled (1 - 4)
value	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

channel	The Open Drain Pin to be toggled (1 - 4)
value	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 an 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

meterspersecond	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

frequency	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

percentage	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

kg	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

percentage	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 equivilant 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.

```
#pragma once
#include <string>
#include <iostream>
```

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

using json = nlohmann::json;

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

// I2C
#define I2C_ADDRESS 0x50
#define I2C_DEVICE "/dev/i2c-1"

// MODBUS
#define SIWAREX_ADDRESS 0x14

// LED Pins
#define PIN_LED_READY 23
#define PIN_LED_BUSY 24
#define PIN_LED_FAULT 25

// PWM Pin
#define PWM_PIN 13

// 24V Power Switch Pins
#define PIN_POWERSW1 4
#define PIN_POWERSW2 26

// Pins for Impedance switching
#define PIN_IMPEDANCE1 5
#define PIN_IMPEDANCE2 6

// Pins for extra switches (e.g. WebServer, WriteProtect)
#define PIN_EXTRASW1 27
#define PIN_EXTRASW2 22

// Analog Channels
#define ADDVOL_CHANNEL 2
#define SUBVOL_CHANNEL 3
#define CELL_DC 1
#define SEN_OUT 4
#define EXC_IN 1
#define SEN_IN 2

enum LoadCellMode
{
    NORMAL = 0x00,
    OVERLOAD = 0x01,
    INVERTED = 0x02,

} typedef LoadCellMode;

enum IMPEDANCE
{
    OPEN = 0x00,
    NOMINAL = 0x01,
    SHORT = 0x02,

} typedef IMPEDANCE;

enum SYSTEM_TYPE
{
    DOSING_SCALE = 0x01,
    BELT_SCALE = 0x02,
} typedef SYSTEM_TYPE;

enum MATERIAL_FLOW
{
    NONE = 0x00,
    EMPTY,
    FINE,
    COARSE,
    XCOARSE,

} typedef MATERIAL_FLOW;

enum RUN_MODE
{
    AUTO,
    PASSIVE,
    MANUAL,
    IDLE,
    OFF

} typedef RUN_MODE;

struct CUBIC_FUNCTION
{
    float a, b, c, d;
} typedef CUBIC_FUNCTION;

class Configuration
{
public:
    Configuration(std::string path);
    ~Configuration();

    void loadConfiguration();

    // SETTING VARIABLES
    LoadCellMode cellMode = LoadCellMode::NORMAL;
    SYSTEM_TYPE systemType = SYSTEM_TYPE::DOSING_SCALE;
    float exc_voltage = 10.f;
    float load_weight = 20.f;
    float initial_weight = 10.f;
    float addvol_ratio = 500;
    float max_diff_voltage = 40;
    float cellCharecteristic = 4;
    float speedAt100 = 5;
    float freqAt100 = 10000;

    float startVoltage = 2;
    float endVoltage = 9;

    float a, b, c, d;
    CUBIC_FUNCTION calibrationReg;

    // Input channel assignment
    MATERIAL_FLOW inputChannel1 = MATERIAL_FLOW::EMPTY;
    MATERIAL_FLOW inputChannel2 = MATERIAL_FLOW::FINE;
    MATERIAL_FLOW inputChannel3 = MATERIAL_FLOW::COARSE;
    MATERIAL_FLOW inputChannel4 = MATERIAL_FLOW::XCOARSE;

private:
    void parseJSON();
    std::string _path;
};
```

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.

```
#pragma once
#include <signal.h>
#include <pigpio.h>
#include <stdint.h>
```

```
#include <cstdio>

class GPIO
{
public:
    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);

private:
};
```

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.

```
#pragma once
#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>

class I2C
{
public:
    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();

private:
    std::string _dev;
    uint16_t _address;
    int i2c0 = -1;
};
```

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.

```
#pragma once
#include "Configuration.hpp"
#include "I2C.hpp"
#include "utility.hpp"

#include <chrono>
#include <thread>
#include <iostream>
using namespace std::chrono_literals;

enum TRANSITION
{
   DISABLE = 0x00,
   RISING = 0x01,
   FALLING = 0x02,
   BOTH = 0x03,
   UNDEFINED = 0x04
} typedef TRANSITION;

class IABoard
{
public:
   IABoard();
   ~IABoard();

   // Check if the board is responding
   bool detectBoard();

   // Read all digital inputs
   uint8_t digitalRead();
   // Read digital input of certain channel 1 - 4
   bool digitalRead(uint8_t channel);
   bool getDigitalRead(uint8_t channel);

   // Reads the number of counted transitions (if enabled)
   uint16_t readTransitions(uint8_t channel);
   // Reads th ecurrently set transition type
   TRANSITION getTransistionType(uint8_t channel);
   // Sets the type of transistions that should be counted
   void setTransistionType(uint8_t channel, TRANSITION tran);
   // Sets the transistion counter of a channel to 0
   void resetTransitions(uint8_t channel);

   // Get the currently set analog output voltage
   float getAnalogVolOut(uint8_t channel);
   // Set the analog output voltage from 0 - 10V, voltage in volts
   void setAnalogVolOut(uint8_t channel, float voltage);

   // Get the currently set analog output current
   float getAnalogCurOut(uint8_t channel);
   // Set the analog output current from 4 - 20mA, current in mA
   void setAnalogCurOut(uint8_t channel, float current);

   // Get the PWM Duty Cycle for the Open Drain Output (if not used as digital out)
   float getOpenDrainPWM(uint8_t channel);
   // Set the PWM Duty Cycle (0 - 100%) for the Open Drain Output
   void setOpenDrainPWM(uint8_t channel, float dutyCycle);

   // Read all digital open drain outputs
   uint8_t getOpenDrainDOUT();
   // Get the currently set open drain digital out value
   bool getOpenDrainDOUT(uint8_t channel);
   // Set the digital open drain output
   void setOpenDrainDOUT(uint8_t channel, bool value);

   // Gets the state of a certain LED
   bool getLED(uint8_t channel);
   // Sets a certain LED Low or High
   void setLED(uint8_t channel, bool value);
   // Sets all LEDs ON or OFF
   void setAllLED(bool value);

   // Reads the analog input voltage of a certain channel (0-10V)
   float readAnalogVolIn(uint8_t channel);
   // Reads the analog input voltage of a certain channel (-10-10V, Jumper set)
   float readAnalogVolInPM(uint8_t channel);

   // Reads the analog input current of a certain channel (4-20mA)
   float readAnalogCurIn(uint8_t channel);

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

```

    // Turn all digital and analog outputs off
    void setAllOFF();

private:
    I2C *_i2c;

    bool _digitalRead[4] = {0, 0, 0, 0};

    uint8_t _fwVersion[2] = {0x00, 0x00};
    uint8_t _boardTemperature = 0;
    float _24Vrail = 0.f;
    float _5Vrail = 0.f;

    // Delay because the IA-Board can only handle commands every few ms
    const std::chrono::milliseconds _delayBetweenCommands = 2ms;
    std::chrono::time_point<std::chrono::system_clock, std::chrono::duration<double» _lastCommand;

    // Wait till the minimum time between commands has elapsed
    void waitForIA();
};
```

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.

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

struct CURVE
{
    float startDelay = 0;
    float stopDelay = 0;
    float riseTime = 1;
    float fallTime = 1;
    float maxFlow = 1;
} typedef CURVE;

class MaterialFlow
{
public:
    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);

private:
    uint8_t _channel;
    MATERIAL_FLOW _flowType;

    bool _lastPinState = 0;
    float _currentFlow = 0;
    float _lastPinStateTime = 0;

    CURVE _curve;

    IABoard *_ia;
};
```

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.

```
#pragma once
#include <vector>
#include <iostream>
#include "Configuration.hpp"

#include "UART.hpp"

class Modbus
{
public:
    Modbus();
    ~Modbus();

    void transmitRequest(uint16_t startRegister, uint16_t length);
    std::vector<uint8_t> receiveResponse();

private:
    uint16_t calculateCRC(uint8_t *data, int length);

    UART *_uart;
    const uint8_t _address = SIWAREX_ADDRESS;
};
```

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.

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

class PCB
{
public:
   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();

private:
   GPIO *_gpio;
   IABoard *_ia;
   Configuration *_config;
};
```

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.

```
#pragma once
#include "PCB.hpp"
#include "Configuration.hpp"
#include "IABoard.hpp"
#include "MaterialFlow.hpp"
#include "SIWAREX.hpp"

#include <chrono>
#include <thread>

using namespace std::chrono_literals;

class Simulator
{
public:
    Simulator();
    ~Simulator();

    void setWeightPER(float percentage); // Set the weight from 0 - 100% of nominal Load
    void setWeightKG(float kg);          // Set the weight in 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);

private:
    Configuration *_config;
    PCB *_pcb;
    IABoard *_ia;
    SIWAREX *_siwarex;

    MaterialFlow *_materialFlows[4];
};
```

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.

```
#pragma once
#include <vector>

#include "SIWAREX_REGISTER.hpp"
#include "Modbus.hpp"
#include "utility.hpp"

class SIWAREX
{
public:
    SIWAREX();
    ~SIWAREX();

    float getLoadcellVoltage();

private:
    float requestFloat(uint16_t startRegister);

    Modbus *_modbus;
};
```

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.

```
#pragma once
#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.

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

class UART
{
public:
   UART();
   ~UART();
   bool begin();
   bool transmitMSG(uint8_t *msg, uint16_t length);
   std::vector<uint8_t> receiveMSG();

private:
   int uart0 = -1;
   // std::string _dev;
   const uint8_t _messageSizeRX = 0;    // Number of bytes to wait for
   const uint8_t _messageTimeoutRX = 1; // Read Timeout in 0.1s steps
};
```

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

values	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

a	Coefficient in front of x^3
b	Coefficient in front of x^2
c	Coefficient in front of x^1
d	Coefficient in front of x^0
x	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 $f'(x) = 3ax^2+2bx+cx$

Parameters

a	Coefficient in front of x^3
b	Coefficient in front of x^2
c	Coefficient in front of x^1
x	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

value	The value to be clipped
max	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

value	The value to be clipped
min	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

value	The value to be clipped
min	The lower limit
max	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

x	Vector of x-values of the dataset
y	Vector of y-values of the dataset
a	Coefficient in front of x^3
b	Coefficient in front of x^2
c	Coefficient in front of x^1
d	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

x	Vector of x-values of the dataset
y	Vector of y-values of the dataset
m	Pointer to the slope m
b	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

a	Coefficient in front of x^3
b	Coefficient in front of x^2
c	Coefficient in front of x^1
d	Coefficient in front of x^0
value	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 *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)

4.35.1 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

values	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

a	Coefficient in front of x^3
b	Coefficient in front of x^2
c	Coefficient in front of x^1
d	Coefficient in front of x^0
x	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 $f'(x) = 3ax^2+2bx+cx$

Parameters

a	Coefficient in front of x^3
b	Coefficient in front of x^2
c	Coefficient in front of x^1
x	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

value	The value to be clipped
max	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

value	The value to be clipped
min	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

value	The value to be clipped
min	The lower limit
max	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

x	Vector of x-values of the dataset
y	Vector of y-values of the dataset
a	Coefficient in front of x^3
b	Coefficient in front of x^2
c	Coefficient in front of x^1
d	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

x	Vector of x-values of the dataset
y	Vector of y-values of the dataset
m	Pointer to the slope m
b	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

a	Coefficient in front of x^3
b	Coefficient in front of x^2
c	Coefficient in front of x^1
d	Coefficient in front of x^0
value	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.

```
#pragma once
#include <vector>
#include <iostream>
#include <cmath>
#include <chrono>
#include <thread>

using namespace std::chrono_literals;

#include "Eigen/Dense"

using Eigen::MatrixXd;
using Eigen::VectorXd;

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);
```

Index