# FunTechHouse RoomTemperature

Generated by Doxygen 1.8.4

Sat Mar 8 2014 15:36:53

# Contents

Chapter 1

# The FunTechHouse RoomTemperature

Room temperature sensor for the FunTechHouse project.This project uses a Arduino with a Ethernet shield, and sends its results using MQTT to a Mosquitto server.

**See Also**

`http://fun-tech.se/FunTechHouse/RoomTemperature/`
`https://github.com/jsiei97/FunTechHouse_RoomTemperature`

# Chapter 2

# Todo List

**Member TemperatureSensor::init (int pin, FT_SensorType type)**
Make sure it is called only once, or fix multiple new.

# Chapter 3

# Hierarchical Index

## 3.1 Class Hierarchy

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

# Chapter 4

# Class Index

## 4.1 Class List

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

# Chapter 5

# File Index

## 5.1 File List

Here is a list of all documented files with brief descriptions:

# Chapter 6

# Class Documentation

## 6.1 DS18B20 Class Reference

OneWire DS18B20 Temperature Sensor Class.

### Public Member Functions

- DS18B20 (int pin)

  *Init with what pin the sensor is connected to.*
- bool getTemperature (double ∗value)

  *Returns a temperature from a OneWire sensor.*

### 6.1.1 Detailed Description

OneWire DS18B20 Temperature Sensor Class.

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 DS18B20::DS18B20 ( int *pin* )

Init with what pin the sensor is connected to.

**Parameters**

| | | |
|---|---|---|
| `in` | *pin* | is the IO pin |

### 6.1.3 Member Function Documentation

#### 6.1.3.1 bool DS18B20::getTemperature ( double ∗ *value* )

Returns a temperature from a OneWire sensor.

Please note that there is a need for a 750ms delay in the middle that is removed so this function starts a new reading and returns the result from the last reading.

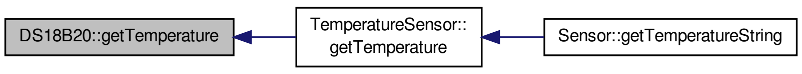
**Parameters**

| | | |
|---|---|---|
| out | *value* | Temperature reading |

**Returns**

true if ok, false if fail.

Here is the caller graph for this function:


DS18B20::getTemperature
TemperatureSensor::getTemperature
Sensor::getTemperatureString

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

- FunTechHouse_RoomTemperature/DS18B20.h
- FunTechHouse_RoomTemperature/DS18B20.cpp

## 6.2 LVTS Class Reference

Low Voltage Temperature Sensor Class.

### Public Member Functions

- LVTS (int pin, FT_SensorType type)

    *Init with IO pin and sensor type.*
- bool getTemperature (double ∗value)

    *Get the current temperature from this sensor.*

### 6.2.1 Detailed Description

Low Voltage Temperature Sensor Class.

This is a wrapper class for analog low voltage temperature sensors like the LM35 and LM34.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 LVTS::LVTS ( int *pin*, FT_SensorType *type* )

Init with IO pin and sensor type.

**Parameters**

| | | |
|---|---|---|
| in | *pin* | is the IO pin |
| in | *type* | is sensor type |

### 6.2.3 Member Function Documentation

#### 6.2.3.1 bool LVTS::getTemperature ( double ∗ *value* )

Get the current temperature from this sensor.

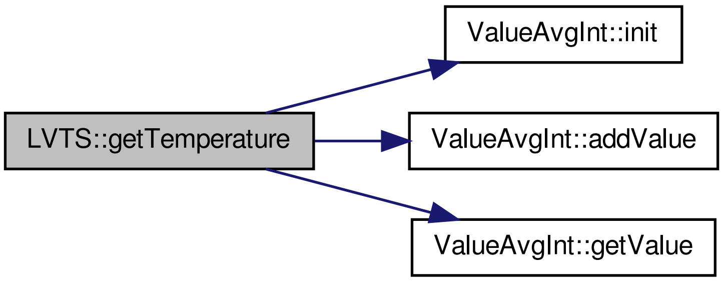
**Parameters**

| | | |
|---|---|---|
| `out` | *value* | is the temperature return value |

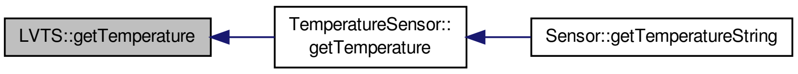
**Returns**

true if ok

Here is the call graph for this function:



Here is the caller graph for this function:


LVTS::getTemperature
TemperatureSensor::getTemperature
Sensor::getTemperatureString

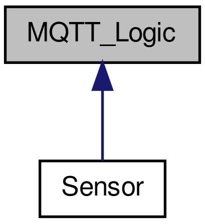
The documentation for this class was generated from the following files:

- FunTechHouse_RoomTemperature/LVTS.h
- FunTechHouse_RoomTemperature/LVTS.cpp

## 6.3 MQTT_Logic Class Reference

The MQTT logic functions that can be inherited.

Inheritance diagram for MQTT_Logic:



### Public Member Functions

- MQTT_Logic ()

  *Default constructur.*
- bool setTopic (char ∗topicSubscribe, char ∗topicPublish)

  *What mqtt topics this sensor will use.*
- char ∗ getTopicSubscribe ()

  *Get the stored subscribe topic.*
- char ∗ getTopicPublish ()

  *Get the stored publish topic.*
- bool checkTopicSubscribe (char ∗check)

  *Is this topic the same as the stored one?*

## 6.3.1 Detailed Description

The MQTT logic functions that can be inherited.

## 6.3.2 Member Function Documentation

### 6.3.2.1 bool MQTT_Logic::checkTopicSubscribe ( char ∗ *check* )

Is this topic the same as the stored one?

**Parameters**

| | | |
|---|---|---|
| in | *check* | string to compare with |

**Returns**

true if same, false if not the same.

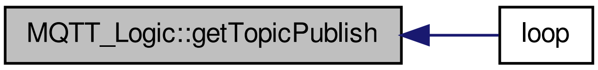
### 6.3.2.2 char ∗ MQTT_Logic::getTopicPublish ( )

Get the stored publish topic.

**Returns**

the stored string

Here is the caller graph for this function:



#### 6.3.2.3 char ∗ MQTT_Logic::getTopicSubscribe ( )

Get the stored subscribe topic.

**Returns**

the stored string

#### 6.3.2.4 bool MQTT_Logic::setTopic ( char ∗ *topicSubscribe,* char ∗ *topicPublish* )

What mqtt topics this sensor will use.

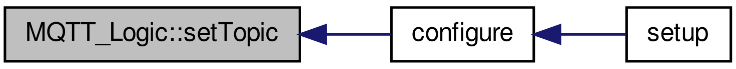
**Parameters**

| | | |
|---|---|---|
| `in` | *topicSubscribe* | data from the mqtt server |
| `in` | *topicPublish* | data to the mqtt server |

**Returns**

true if ok

Here is the caller graph for this function:



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

- FunTechHouse_RoomTemperature/MQTT_Logic.h
- FunTechHouse_RoomTemperature/MQTT_Logic.cpp

## 6.4 OneWire Class Reference

### Public Member Functions

- **OneWire** (uint8_t pin)
- uint8_t **reset** (void)
- void **select** (const uint8_t rom[8])
- void **skip** (void)
- void **write** (uint8_t v, uint8_t power=0)
- void **write_bytes** (const uint8_t ∗buf, uint16_t count, bool power=0)
- uint8_t **read** (void)
- void **read_bytes** (uint8_t ∗buf, uint16_t count)
- void **write_bit** (uint8_t v)
- uint8_t **read_bit** (void)
- void **depower** (void)
- void **reset_search** ()
- void **target_search** (uint8_t family_code)
- uint8_t **search** (uint8_t ∗newAddr)

### Static Public Member Functions

- static uint8_t **crc8** (const uint8_t ∗addr, uint8_t len)
- static bool **check_crc16** (const uint8_t ∗input, uint16_t len, const uint8_t ∗inverted_crc, uint16_t crc=0)
- static uint16_t **crc16** (const uint8_t ∗input, uint16_t len, uint16_t crc=0)

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

- FunTechHouse_RoomTemperature/OneWire.h
- FunTechHouse_RoomTemperature/OneWire.cpp

## 6.5 PubSubClient Class Reference

### Public Member Functions

- **PubSubClient** (uint8_t ∗, uint16_t, void(∗)(char ∗, uint8_t ∗, unsigned int))
- **PubSubClient** (char ∗, uint16_t, void(∗)(char ∗, uint8_t ∗, unsigned int))
- boolean **connect** (char ∗)
- boolean **connect** (char ∗, char ∗, uint8_t, uint8_t, char ∗)
- void **disconnect** ()
- boolean **publish** (char ∗, char ∗)
- boolean **publish** (char ∗, uint8_t ∗, unsigned int)
- boolean **publish** (char ∗, uint8_t ∗, unsigned int, boolean)
- boolean **subscribe** (char ∗)
- boolean **loop** ()
- boolean **connected** ()

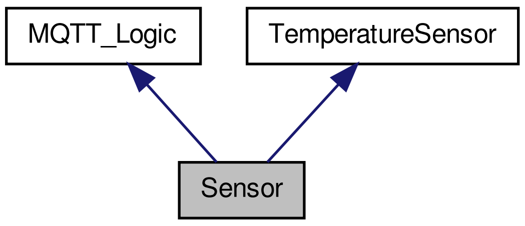
The documentation for this class was generated from the following files:

- FunTechHouse_RoomTemperature/PubSubClient.h
- FunTechHouse_RoomTemperature/PubSubClient.cpp

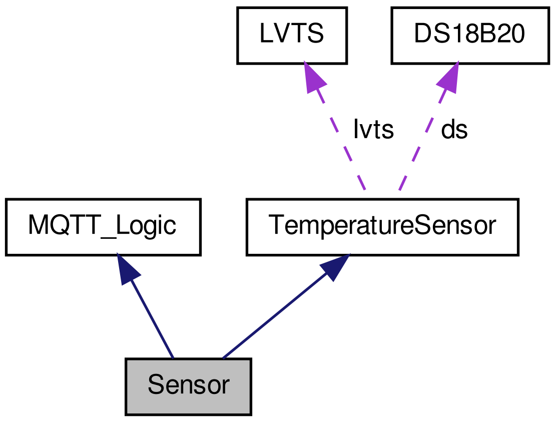
## 6.6 Sensor Class Reference

A temperature sensor class with alarm logic.

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:



### Public Member Functions

- bool getTemperatureString (char ∗str, int size)

  *Get temperature in a mqtt formatted string.*
- SensorAlarmNumber alarmCheckString (char ∗str, int size)

  *Get the alarm in a mqtt formatted string.*

### Additional Inherited Members

### 6.6.1 Detailed Description

A temperature sensor class with alarm logic.

### 6.6.2 Member Function Documentation

#### 6.6.2.1 SensorAlarmNumber Sensor::alarmCheckString ( char ∗ *str,* int *size* )

Get the alarm in a mqtt formatted string.

It is ok to call this function until it returns SENSOR_ALARM_NO.

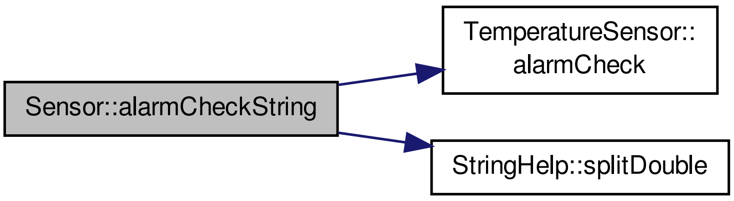
**Parameters**

| | | |
|---|---|---|
| out | *str* | returns a string with alarm data |
| in | *size* | The string max size |

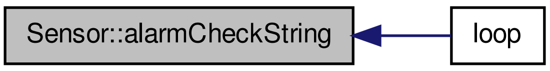
**Returns**

SensorAlarmNumber what alarm is active.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.6.2.2 bool Sensor::getTemperatureString ( char ∗ *str,* int *size* )

Get temperature in a mqtt formatted string.

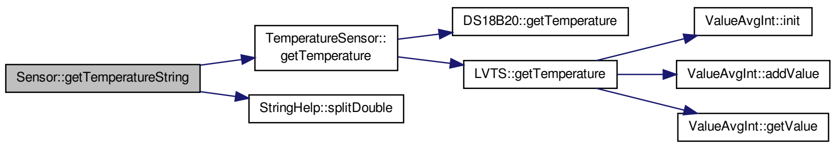
**Parameters**

| | | |
|---|---|---|
| out | *str* | returns a string with temperature data |
| in | *size* | The string max size |

**Returns**

true if ok and it is time to send the data

Here is the call graph for this function:



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

- FunTechHouse_RoomTemperature/Sensor.h
- FunTechHouse_RoomTemperature/Sensor.cpp

## 6.7 StringHelp Class Reference

String helper functions.

### Static Public Member Functions

- static void splitDouble (double value, int ∗integer, int ∗decimal)

  *Split a double into the integer and decimal part, since the arduino sprintf cant handle double.*

### 6.7.1 Detailed Description

String helper functions.

### 6.7.2 Member Function Documentation

#### 6.7.2.1 void StringHelp::splitDouble ( double *value*, int ∗ *integer*, int ∗ *decimal* ) [static]

Split a double into the integer and decimal part, since the arduino sprintf cant handle double.

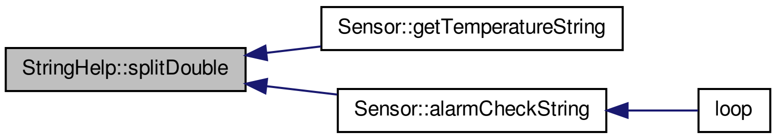
Please note that the decimal part will be 2 digits, so you may need to fill with zero when it is printed. i.e. 4.04 will return 4 and 4, and 5.2 will return 5 and 20.

**Parameters**

| | | |
|---|---|---|
| in | *value* | The value to split |

| | | |
|---|---|---|
| out | *integer* | The integer part that will be returned |
| out | *decimal* | The decimal part that will be returned |

Here is the caller graph for this function:



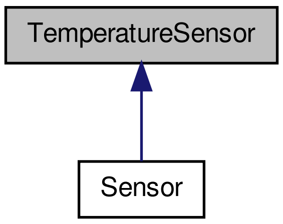
The documentation for this class was generated from the following files:

- FunTechHouse_RoomTemperature/StringHelp.h
- FunTechHouse_RoomTemperature/StringHelp.cpp

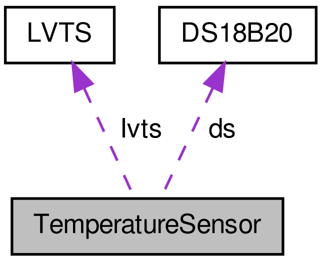
## 6.8 TemperatureSensor Class Reference

A Temperature sensor class for the DS18B20 and LVTS.

Inheritance diagram for TemperatureSensor:



Collaboration diagram for TemperatureSensor:



## Public Member Functions

- void init (int pin, FT_SensorType type)

  *Init this object.*
- void setAlarmLevels (double alarmHyst, bool activateLowAlarm, double alarmLevelLow, bool activateHigh-Alarm, double alarmLevelHigh)

  *Active alarm and set what alarm levels to be used.*
- void setValueDiff (double diff)

  *Enable value diff to send value.*
- void setValueMaxCnt (int cnt)

  *Enable send after X counts even if value is the same.*
- void setValueOffset (double offset)

  *Calibration offset value to add to value.*
- bool getTemperature (double ∗value)

  *Get the current temperature from this sensor.*
- SensorAlarmNumber alarmCheck ()

  *Check if there is any active alarms.*
- void alarmAck (SensorAlarmNumber num)

  *Acknowledge alarm, dont send any more at this time.*

## Protected Attributes

- FT_SensorType type

  *What kind of sensor is this object?*
- DS18B20 ∗ ds

  *Ref to DS18B20 if correct type.*
- LVTS ∗ lvts

  *Ref to LVTS if correct type.*
- AlarmStates alarmSensor

  *Current state for the sensor read alarm.*
- AlarmStates alarmLow

  *Current state for the low level alarm.*
- AlarmStates alarmHigh

  *Current state for the high level alarm.*

- bool alarmLowActive

  *Is Alarm Low activated?*
- bool alarmHighActive

  *Is Alarm High activated?*
- unsigned int failcnt

  *If sensor read fails, then this value inc. Zero is ok.*
- double alarmHyst

  *Hysteresis used to reset the alarm levels.*
- double alarmHighLevel

  *Alarm level for the high value alarm.*
- double alarmLowLevel

  *Alarm level for the low value alarm.*
- double valueWork

  *Active value that we work with right now.*
- double valueSent

  *Last value sent to the server.*
- double valueDiffMax

  *Value should diff more than this to be sent to the server.*
- int valueSendCnt

  *Always send after "cnt time" even if there is no change, the cnt variable.*
- int valueSendMax

  *Always send after "cnt time" even if there is no change, the max value.*
- double valueOffset

  *Offset calibration value, this will just be added to the messured value.*

### 6.8.1 Detailed Description

A Temperature sensor class for the DS18B20 and LVTS.

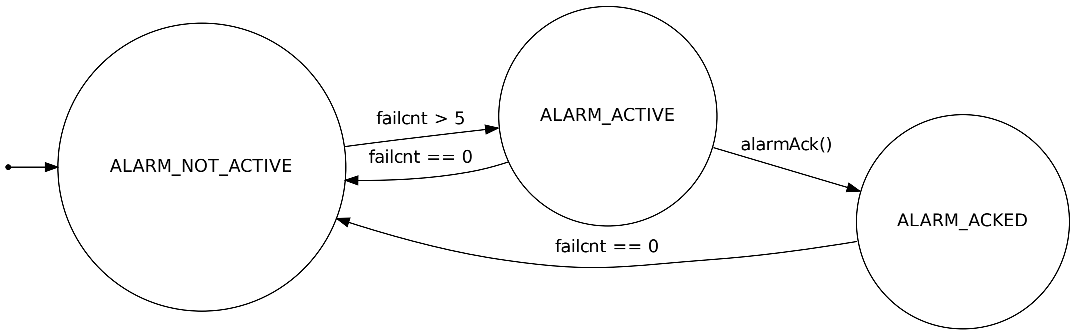
This will class will wrap the different sensors and give them the same interface so they can be put in a array and just looped from main.

There is also some some alarm logic so that main know when things is wrong. The alarm is active as long as it is not ack:ed (Acknowledged) or until what triggered the alarm ends, like the temperature goes back to normal.

Sensor read alarm is triggered if there is a read error.

**See Also**

SENSOR_ALARM_SENSOR

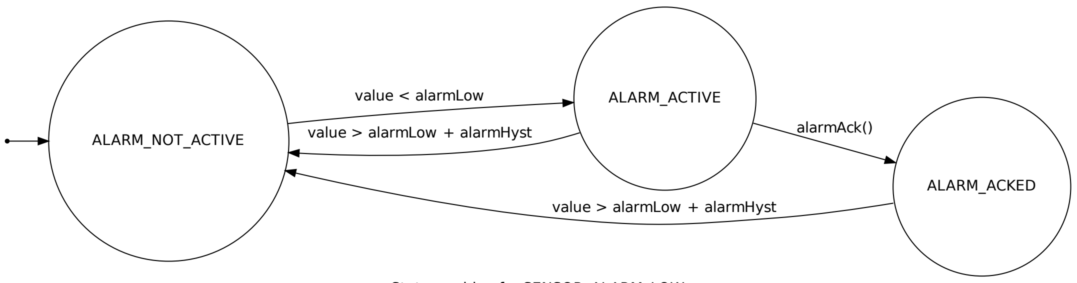


State machine for SENSOR_ALARM_SENSOR

Low level alarm is if the value is lower than the alarm low level.

**See Also**

SENSOR_ALARM_LOW

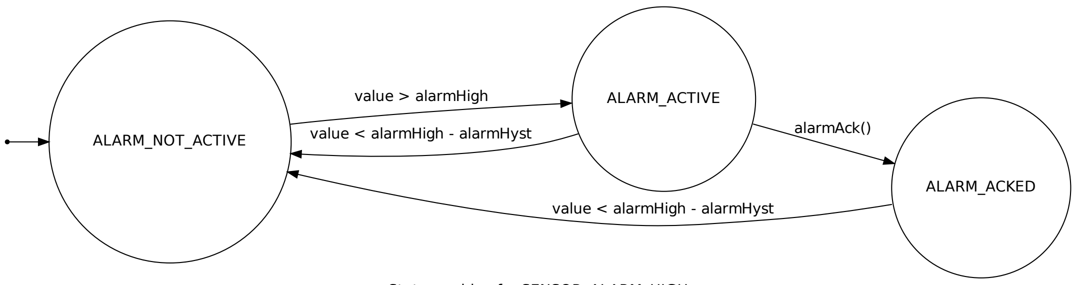


State machine for SENSOR_ALARM_LOW

High level alarm is if the value is higher than alarm high level.

**See Also**

SENSOR_ALARM_HIGH



State machine for SENSOR_ALARM_HIGH

## 6.8.2 Member Function Documentation

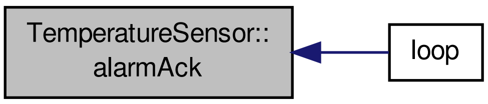
### 6.8.2.1 void TemperatureSensor::alarmAck ( SensorAlarmNumber *num* )

Acknowledge alarm, dont send any more at this time.

**Parameters**

| | |
|---|---|
| *num* | The alarm to ack |

Here is the caller graph for this function:



#### 6.8.2.2 **SensorAlarmNumber** TemperatureSensor::alarmCheck ( )

Check if there is any active alarms.

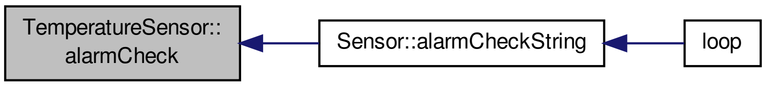
Please note that this should be called after getTemperature().

$see getTemperature

**Returns**

SensorAlarmNumber for the type of alarm.

Here is the caller graph for this function:



#### 6.8.2.3 bool TemperatureSensor::getTemperature ( double ∗ *value* )

Get the current temperature from this sensor.

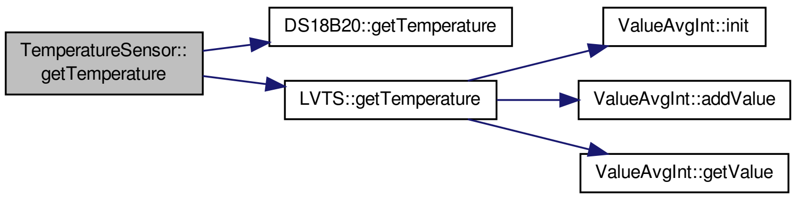
**Parameters**

| | | |
|---|---|---|
| `out` | *value* | is the temperature return value |

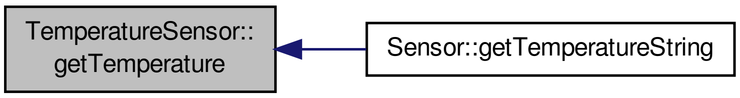
**Returns**

true if ok

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.8.2.4 void TemperatureSensor::init ( int *pin*, FT_SensorType *type* )

Init this object.

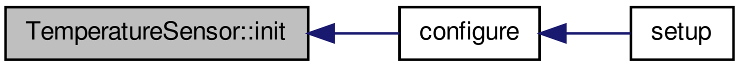
This must only be called once!

**Parameters**

| | | |
|---|---|---|
| in | *pin* | is IO pin |
| in | *type* | is sensor type i.e. DS18B20 or LM35. |

**Todo** Make sure it is called only once, or fix multiple new.

Here is the caller graph for this function:



#### 6.8.2.5 void TemperatureSensor::setAlarmLevels ( double *alarmHyst,* bool *activateLowAlarm,* double *alarmLevelLow,* bool *activateHighAlarm,* double *alarmLevelHigh* )

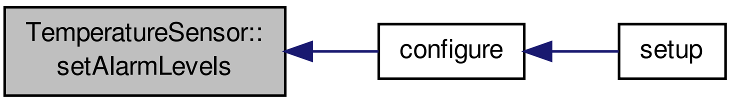
Active alarm and set what alarm levels to be used.

Please note that the low alarm is activated under alarmLowLevel-alarmHyst, and deactivates over alarmLowLevel. The high alarm activates over alarmHighLevel+alarmHyst, and deactivates lower than alarmLowLevel.

**Parameters**

| | |
|---|---|
| *alarmHyst* | How big hysteresis around the alarm level. |
| *activateLow-Alarm* | true to activate low alarm. |
| *alarmLevelLow* | alarm level for low. |
| *activateHigh-Alarm* | true to active high alarm. |
| *alarmLevelHigh* | alarm level for high. |

Here is the caller graph for this function:



#### 6.8.2.6 void TemperatureSensor::setValueDiff ( double *diff* )

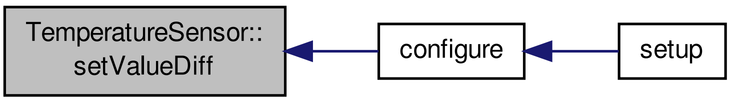
Enable value diff to send value.

Value needs to diff more than this value to be treated as a new value that should be send.

**Parameters**

| | |
|---|---|
| *diff* | the value |

Here is the caller graph for this function:



#### 6.8.2.7 void TemperatureSensor::setValueMaxCnt ( int *cnt* )

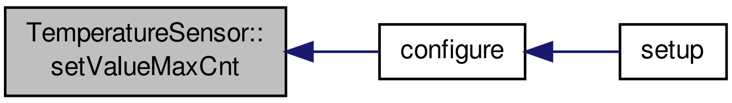
Enable send after X counts even if value is the same.

To make sure that we sometimes get a value even if there is no change.

**Parameters**

| | |
|---|---|
| *cnt* | how many times can we call getTemperature before we always gets a value. |

Here is the caller graph for this function:



#### 6.8.2.8 void TemperatureSensor::setValueOffset ( double *offset* )

Calibration offset value to add to value.

**Parameters**

| | |
|---|---|
| *offset* | a number that will be added onto any read value |

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

- FunTechHouse_RoomTemperature/TemperatureSensor.h
- FunTechHouse_RoomTemperature/TemperatureSensor.cpp

## 6.9 ValueAvgInt Class Reference

A basic filter.

### Public Member Functions

- void init ()

  *Reset and init this filter to start over with a new session.*
- void addValue (int data)

  *Add a new value to the filter.*
- int getValue ()

  *Get the result from the filter.*

### 6.9.1 Detailed Description

A basic filter.

The filter will ignore the most extreme values, and then calculate the average value on the rest.

The usage is first to call init(), then add some 10-20 values with addValue(int). And the result can be colleced with getValue().

### 6.9.2 Member Function Documentation

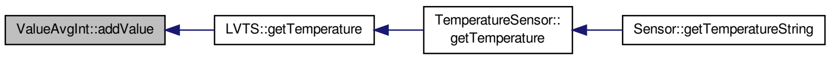
#### 6.9.2.1 void ValueAvgInt::addValue ( int *data* )

Add a new value to the filter.

**Parameters**

|  |  |  |
|---|---|---|
| `in` | *data* | is some data to be used int the filter |

Here is the caller graph for this function:


ValueAvgInt::addValue
LVTS::getTemperature
TemperatureSensor::getTemperature
Sensor::getTemperatureString

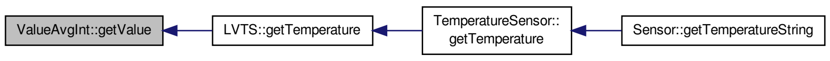
#### 6.9.2.2 int ValueAvgInt::getValue ( )

Get the result from the filter.

**Returns**

the calculated value

Here is the caller graph for this function:


ValueAvgInt::getValue
LVTS::getTemperature
TemperatureSensor::getTemperature
Sensor::getTemperatureString

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

- FunTechHouse_RoomTemperature/ValueAvgInt.h
- FunTechHouse_RoomTemperature/ValueAvgInt.cpp

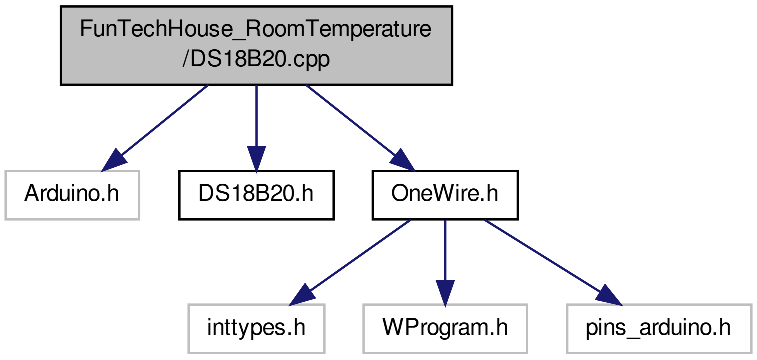
# Chapter 7

# File Documentation

## 7.1 FunTechHouse_RoomTemperature/DS18B20.cpp File Reference

OneWire DS18B20 Temperature Sensor Class.

Include dependency graph for DS18B20.cpp:



### 7.1.1 Detailed Description

OneWire DS18B20 Temperature Sensor Class.

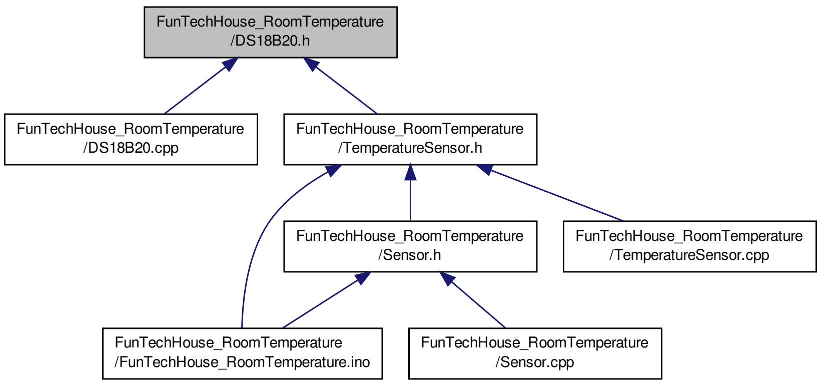
**Author**

Johan Simonsson

## 7.2 FunTechHouse_RoomTemperature/DS18B20.h File Reference

OneWire DS18B20 Temperature Sensor Class.

This graph shows which files directly or indirectly include this file:


FunTechHouse_RoomTemperature /DS18B20.h
FunTechHouse_RoomTemperature /DS18B20.cpp
FunTechHouse_RoomTemperature /TemperatureSensor.h
FunTechHouse_RoomTemperature /Sensor.h
FunTechHouse_RoomTemperature /TemperatureSensor.cpp
FunTechHouse_RoomTemperature /FunTechHouse_RoomTemperature.ino
FunTechHouse_RoomTemperature /Sensor.cpp

### Classes

- class DS18B20

  *OneWire DS18B20 Temperature Sensor Class.*

### 7.2.1 Detailed Description

OneWire DS18B20 Temperature Sensor Class.

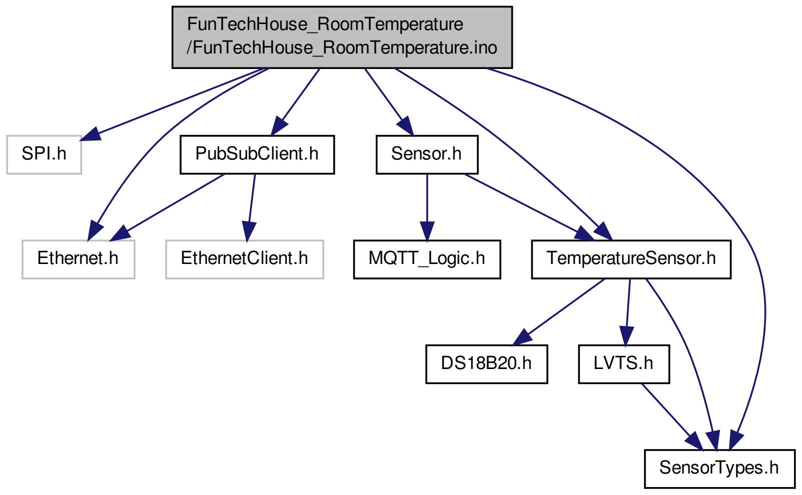
**Author**

Johan Simonsson

## 7.3 FunTechHouse_RoomTemperature/FunTechHouse_RoomTemperature.ino File Reference

Main file.

Include dependency graph for FunTechHouse_RoomTemperature.ino:



### Macros

- #define SENSOR_CNT 2

    *How many sensors shall the sensor array contain.*

- #define OUT_STR_MAX 100

    *Max size for the out string used in the main loop.*

### Functions

- void callback (char ∗topic, uint8_t ∗payload, unsigned int length)

    *The MQTT subscribe callback function.*

- void configure ()

    *Configure this project with device uniq sensor setup.*

- void setup ()

    *First setup, runs once.*

- void loop ()

    *The main loop, runs all the time, over and over again.*

### Variables

- uint8_t mac [] = { 0x90, 0xA2, 0xDA, 0x0D, 0x51, 0xB3 }

  *This device MAC adress, it is written on the Shield and must be uniq.*
- char project_name [] = "FunTechHouse_RoomTemperature"

  *The MQTT device name, this must be unique.*
- Sensor sensor [SENSOR_CNT]

  *The sensor array with active sensors.*
- PubSubClient client ("mosqhub", 1883, callback)

  *The MQTT client.*
- int led = 2

  *Life blink led is connected to IO pin.*

### 7.3.1 Detailed Description

Main file.

**Author**

Johan Simonsson

### 7.3.2 Function Documentation

#### 7.3.2.1 void callback ( char * *topic,* uint8_t * *payload,* unsigned int *length* )

The MQTT subscribe callback function.

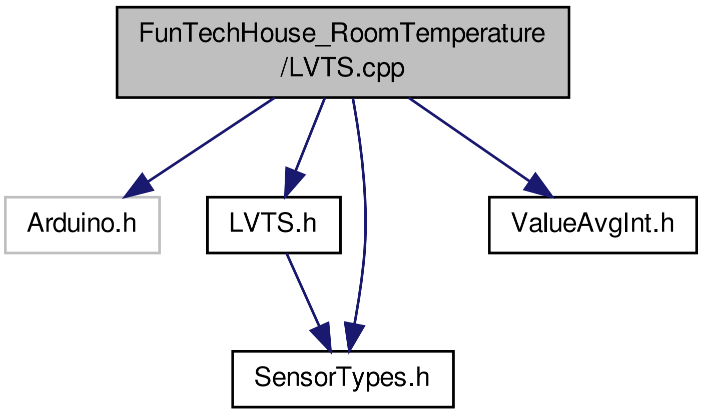
**Parameters**

| | | |
|---|---|---|
| `in` | *topic* | What mqtt topic triggered this callback |
| `in` | *payload* | The actual message |
| `in` | *length* | The message size |

## 7.4 FunTechHouse_RoomTemperature/LVTS.cpp File Reference

Low Voltage Temperature Sensor Class.

Include dependency graph for LVTS.cpp:



### 7.4.1 Detailed Description

Low Voltage Temperature Sensor Class.

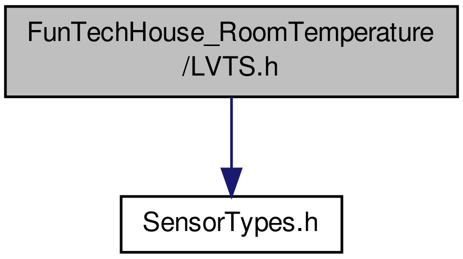
**Author**

Johan Simonsson

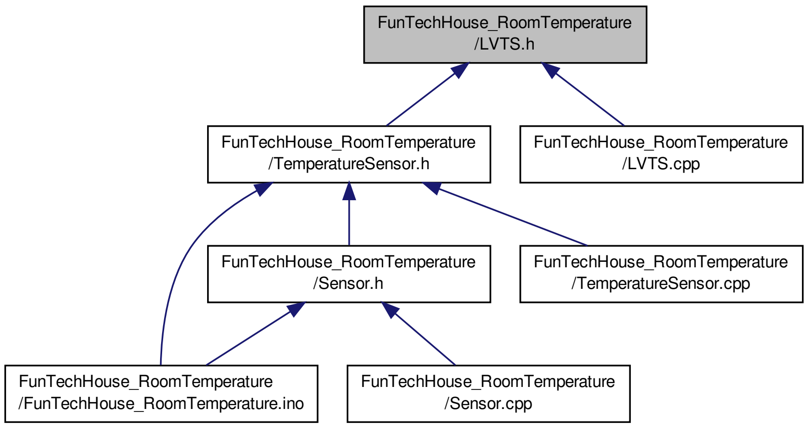
## 7.5 FunTechHouse_RoomTemperature/LVTS.h File Reference

Low Voltage Temperature Sensor Class.

Include dependency graph for LVTS.h:



This graph shows which files directly or indirectly include this file:


FunTechHouse_RoomTemperature /LVTS.h
FunTechHouse_RoomTemperature /TemperatureSensor.h
FunTechHouse_RoomTemperature /LVTS.cpp
FunTechHouse_RoomTemperature /Sensor.h
FunTechHouse_RoomTemperature /TemperatureSensor.cpp
FunTechHouse_RoomTemperature /FunTechHouse_RoomTemperature.ino
FunTechHouse_RoomTemperature /Sensor.cpp

### Classes

- class LVTS

  *Low Voltage Temperature Sensor Class.*

### 7.5.1 Detailed Description

Low Voltage Temperature Sensor Class.

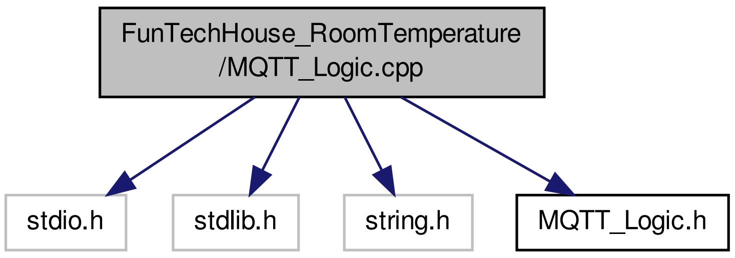
**Author**

Johan Simonsson

## 7.6 FunTechHouse_RoomTemperature/MQTT_Logic.cpp File Reference

The MQTT logic with topics for subscribe and publish.

Include dependency graph for MQTT_Logic.cpp:



### 7.6.1 Detailed Description

The MQTT logic with topics for subscribe and publish.

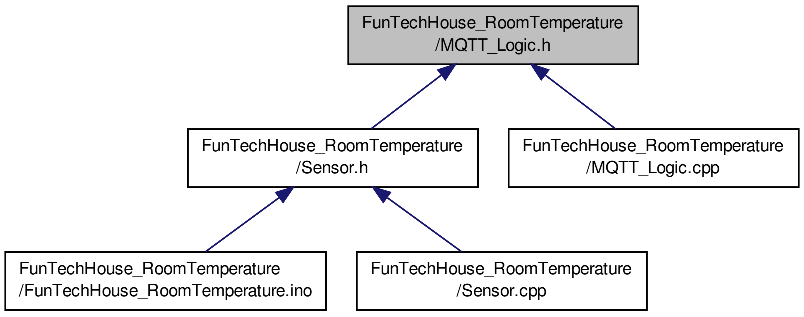
**Author**

Johan Simonsson

## 7.7 FunTechHouse_RoomTemperature/MQTT_Logic.h File Reference

The MQTT logic with topics for subscribe and publish.

This graph shows which files directly or indirectly include this file:


FunTechHouse_RoomTemperature /MQTT_Logic.h
FunTechHouse_RoomTemperature /Sensor.h
FunTechHouse_RoomTemperature /MQTT_Logic.cpp
FunTechHouse_RoomTemperature /FunTechHouse_RoomTemperature.ino
FunTechHouse_RoomTemperature /Sensor.cpp

### Classes

- class MQTT_Logic

  *The MQTT logic functions that can be inherited.*

### 7.7.1 Detailed Description

The MQTT logic with topics for subscribe and publish.

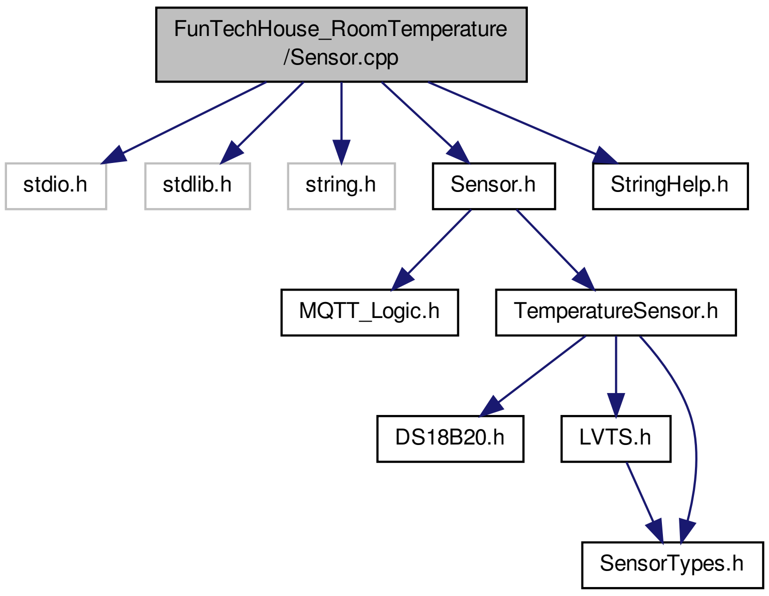
**Author**

Johan Simonsson

## 7.8 FunTechHouse_RoomTemperature/Sensor.cpp File Reference

A temperature sensor class with alarm logic.

Include dependency graph for Sensor.cpp:



### 7.8.1 Detailed Description

A temperature sensor class with alarm logic.

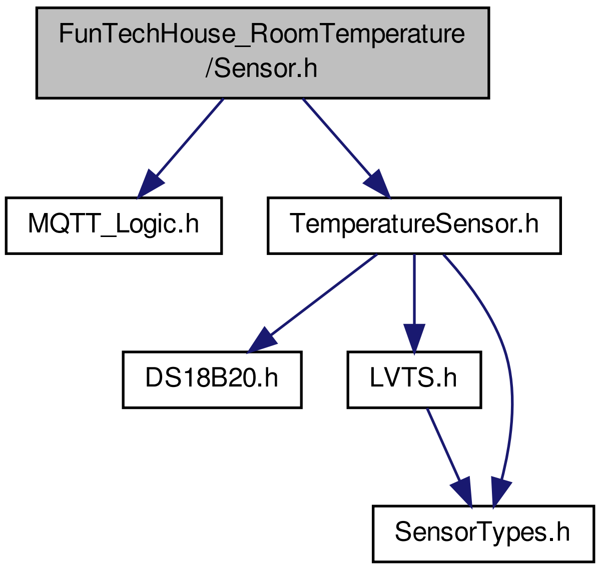
**Author**

Johan Simonsson

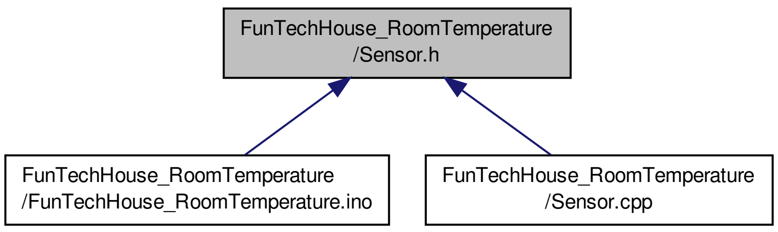
## 7.9 FunTechHouse_RoomTemperature/Sensor.h File Reference

A temperature sensor class with alarm logic.

Include dependency graph for Sensor.h:



This graph shows which files directly or indirectly include this file:


FunTechHouse_RoomTemperature
/Sensor.h
FunTechHouse_RoomTemperature
/FunTechHouse_RoomTemperature.ino
FunTechHouse_RoomTemperature
/Sensor.cpp

### Classes

- class Sensor

  *A temperature sensor class with alarm logic.*

### 7.9.1 Detailed Description

A temperature sensor class with alarm logic.

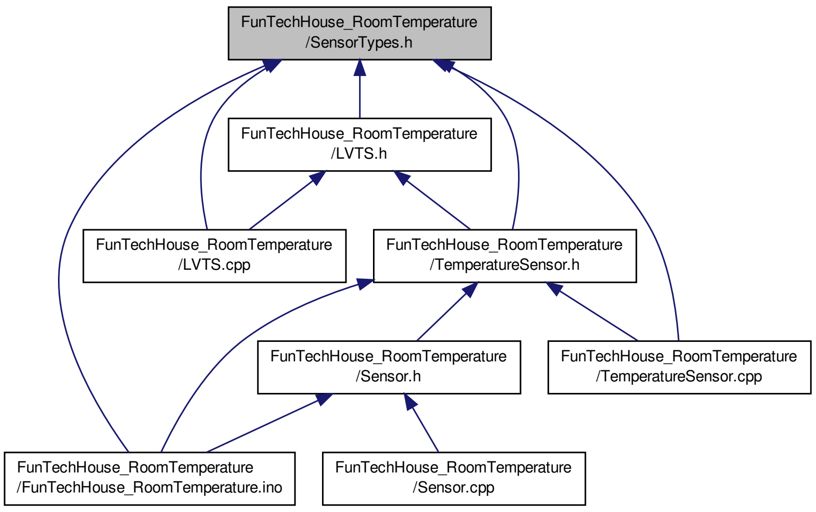
**Author**

Johan Simonsson

## 7.10 FunTechHouse_RoomTemperature/SensorTypes.h File Reference

SensorType has the supported list of sensors.

This graph shows which files directly or indirectly include this file:


FunTechHouse_RoomTemperature
/SensorTypes.h
FunTechHouse_RoomTemperature
/LVTS.h
FunTechHouse_RoomTemperature
/LVTS.cpp
FunTechHouse_RoomTemperature
/TemperatureSensor.h
FunTechHouse_RoomTemperature
/Sensor.h
FunTechHouse_RoomTemperature
/TemperatureSensor.cpp
FunTechHouse_RoomTemperature
/FunTechHouse_RoomTemperature.ino
FunTechHouse_RoomTemperature
/Sensor.cpp

### Enumerations

- enum FT_SensorType { SENSOR_NONE = 0, SENSOR_DS18B20, SENSOR_LVTS_LM34, SENSOR_LV-TS_LM35 }

    *A enumeration of supported sensors.*

### 7.10.1 Detailed Description

SensorType has the supported list of sensors.

**Author**

Johan Simonsson

### 7.10.2 Enumeration Type Documentation

#### 7.10.2.1 enum FT_SensorType

A enumeration of supported sensors.

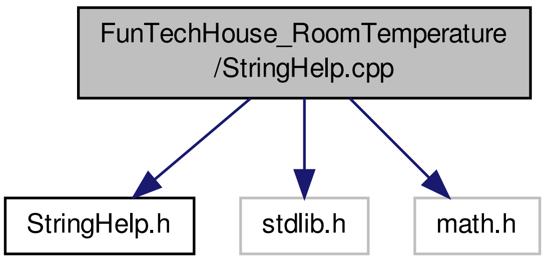
**Enumerator**

| | |
|---|---|
| ***SENSOR_NONE*** | No sensor. |
| ***SENSOR_DS18B20*** | DS18B20 a OneWire temperature sensor. |
| ***SENSOR_LVTS_LM34*** | LM34 a low voltage temperature sensor. 10mV per degF. |
| ***SENSOR_LVTS_LM35*** | LM35 a low voltage temperature sensor. 10mV per degC. |

## 7.11 FunTechHouse_RoomTemperature/StringHelp.cpp File Reference

Helper functions.

Include dependency graph for StringHelp.cpp:



### 7.11.1 Detailed Description

Helper functions.

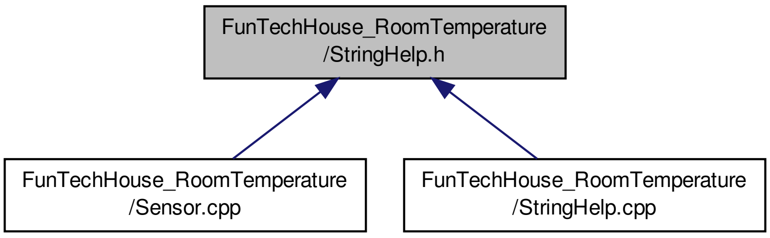
**Author**

Johan Simonsson

## 7.12 FunTechHouse_RoomTemperature/StringHelp.h File Reference

String helper functions.

This graph shows which files directly or indirectly include this file:


FunTechHouse_RoomTemperature /StringHelp.h
FunTechHouse_RoomTemperature /Sensor.cpp
FunTechHouse_RoomTemperature /StringHelp.cpp

### Classes

- class StringHelp

  *String helper functions.*

### 7.12.1 Detailed Description

String helper functions.

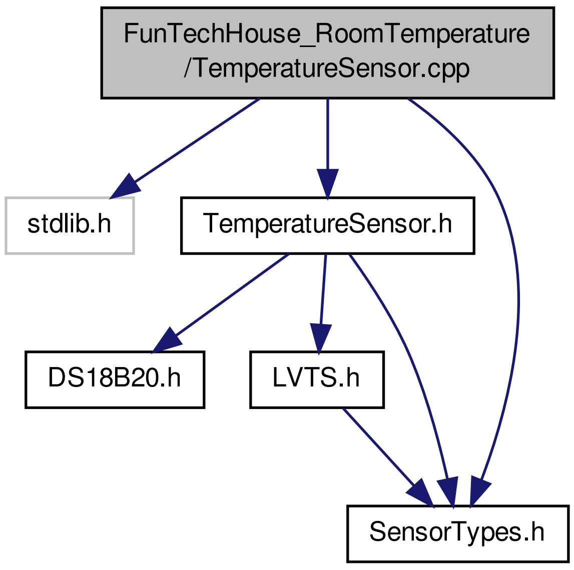
**Author**

Johan Simonsson

## 7.13 FunTechHouse_RoomTemperature/TemperatureSensor.cpp File Reference

A temperature sensor class with alarm logic.

Include dependency graph for TemperatureSensor.cpp:



### 7.13.1 Detailed Description

A temperature sensor class with alarm logic.

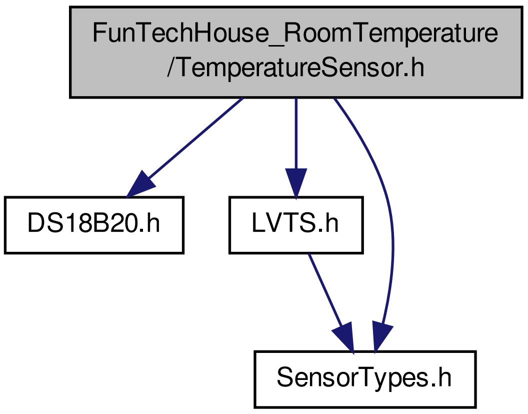
**Author**

Johan Simonsson

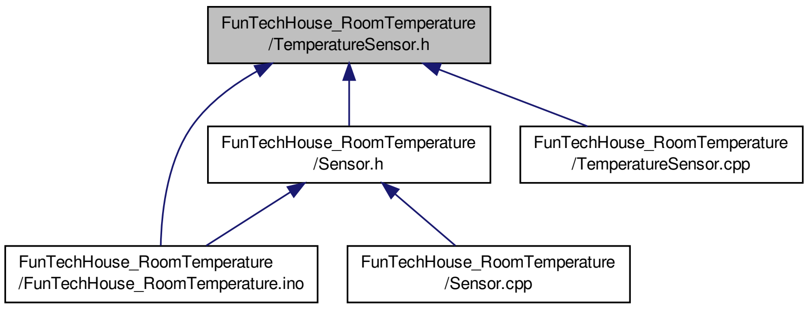
## 7.14 FunTechHouse_RoomTemperature/TemperatureSensor.h File Reference

A temperature sensor class with alarm logic.

Include dependency graph for TemperatureSensor.h:



This graph shows which files directly or indirectly include this file:


FunTechHouse_RoomTemperature
/TemperatureSensor.h
FunTechHouse_RoomTemperature
/Sensor.h
FunTechHouse_RoomTemperature
/TemperatureSensor.cpp
FunTechHouse_RoomTemperature
/FunTechHouse_RoomTemperature.ino
FunTechHouse_RoomTemperature
/Sensor.cpp

### Classes

- class TemperatureSensor

  *A Temperature sensor class for the DS18B20 and LVTS.*

### Enumerations

- enum SensorAlarmNumber { SENSOR_ALARM_NO = 0, SENSOR_ALARM_SENSOR, SENSOR_ALARM-_HIGH, SENSOR_ALARM_LOW }

  *A list with the different alarms.*
- enum AlarmStates { ALARM_NOT_ACTIVE =0, ALARM_ACTIVE, ALARM_ACKED }

  *The statemachine for the alarm.*

## 7.14.1 Detailed Description

A temperature sensor class with alarm logic.

**Author**

Johan Simonsson

## 7.14.2 Enumeration Type Documentation

### 7.14.2.1 enum AlarmStates

The statemachine for the alarm.

**Enumerator**

| | |
|---|---|
| ***ALARM_NOT_ACTIVE*** | The alarm is not triggered, all is fine. |
| ***ALARM_ACTIVE*** | The alarm is triggered. |
| ***ALARM_ACKED*** | The alarm is triggered, and is ack:ed. |

### 7.14.2.2 enum SensorAlarmNumber

A list with the different alarms.

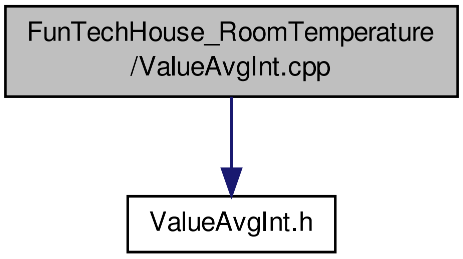
**Enumerator**

| | |
|---|---|
| ***SENSOR_ALARM_NO*** | No active alarm. |
| ***SENSOR_ALARM_SENSOR*** | There is a sensor error. |
| ***SENSOR_ALARM_HIGH*** | High level alarm. |
| ***SENSOR_ALARM_LOW*** | Low level alarm. |

## 7.15 FunTechHouse_RoomTemperature/ValueAvgInt.cpp File Reference

A basic filter.

Include dependency graph for ValueAvgInt.cpp:



### 7.15.1 Detailed Description

A basic filter.

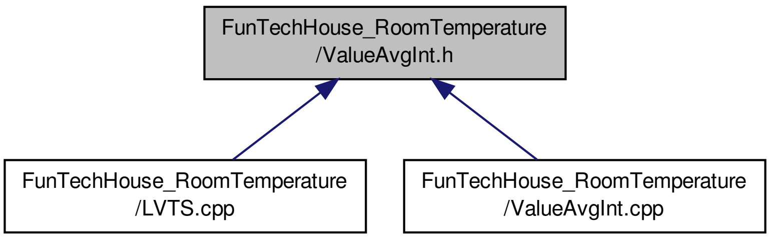
**Author**

Johan Simonsson

## 7.16 FunTechHouse_RoomTemperature/ValueAvgInt.h File Reference

A basic filter.

This graph shows which files directly or indirectly include this file:


FunTechHouse_RoomTemperature
/ValueAvgInt.h
FunTechHouse_RoomTemperature
/LVTS.cpp
FunTechHouse_RoomTemperature
/ValueAvgInt.cpp

### Classes

- class ValueAvgInt

  *A basic filter.*

### 7.16.1 Detailed Description

A basic filter.

**Author**

Johan Simonsson

# Index