

# FunTechHouse RoomTemperature

Generated by Doxygen 1.8.4

Sat Mar 8 2014 15:36:53



# Contents

<b>1</b>	<b>The FunTechHouse RoomTemperature</b>	<b>1</b>
<b>2</b>	<b>Todo List</b>	<b>3</b>
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Class Documentation</b>	<b>11</b>
6.1	DS18B20 Class Reference . . . . .	11
6.1.1	Detailed Description . . . . .	11
6.1.2	Constructor & Destructor Documentation . . . . .	11
6.1.2.1	DS18B20 . . . . .	11
6.1.3	Member Function Documentation . . . . .	11
6.1.3.1	getTemperature . . . . .	11
6.2	LVTS Class Reference . . . . .	12
6.2.1	Detailed Description . . . . .	12
6.2.2	Constructor & Destructor Documentation . . . . .	12
6.2.2.1	LVTS . . . . .	12
6.2.3	Member Function Documentation . . . . .	13
6.2.3.1	getTemperature . . . . .	13
6.3	MQTT_Logic Class Reference . . . . .	14
6.3.1	Detailed Description . . . . .	15
6.3.2	Member Function Documentation . . . . .	15
6.3.2.1	checkTopicSubscribe . . . . .	15
6.3.2.2	getTopicPublish . . . . .	15
6.3.2.3	getTopicSubscribe . . . . .	16
6.3.2.4	setTopic . . . . .	16

6.4	OneWire Class Reference	17
6.5	PubSubClient Class Reference	17
6.6	Sensor Class Reference	18
6.6.1	Detailed Description	19
6.6.2	Member Function Documentation	19
6.6.2.1	alarmCheckString	19
6.6.2.2	getTemperatureString	19
6.7	StringHelp Class Reference	20
6.7.1	Detailed Description	20
6.7.2	Member Function Documentation	20
6.7.2.1	splitDouble	20
6.8	TemperatureSensor Class Reference	21
6.8.1	Detailed Description	23
6.8.2	Member Function Documentation	24
6.8.2.1	alarmAck	24
6.8.2.2	alarmCheck	25
6.8.2.3	getTemperature	25
6.8.2.4	init	26
6.8.2.5	setAlarmLevels	27
6.8.2.6	setValueDiff	27
6.8.2.7	setValueMaxCnt	27
6.8.2.8	setValueOffset	29
6.9	ValueAvgInt Class Reference	29
6.9.1	Detailed Description	29
6.9.2	Member Function Documentation	30
6.9.2.1	addValue	30
6.9.2.2	getValue	31
<b>7</b>	<b>File Documentation</b>	<b>33</b>
7.1	FunTechHouse_RoomTemperature/DS18B20.cpp File Reference	33
7.1.1	Detailed Description	33
7.2	FunTechHouse_RoomTemperature/DS18B20.h File Reference	34
7.2.1	Detailed Description	34
7.3	FunTechHouse_RoomTemperature/FunTechHouse_RoomTemperature.ino File Reference	35
7.3.1	Detailed Description	36
7.3.2	Function Documentation	36
7.3.2.1	callback	36
7.4	FunTechHouse_RoomTemperature/LVTS.cpp File Reference	36
7.4.1	Detailed Description	37
7.5	FunTechHouse_RoomTemperature/LVTS.h File Reference	37

7.5.1 Detailed Description	38
7.6 FunTechHouse_RoomTemperature/MQTT_Logic.cpp File Reference	38
7.6.1 Detailed Description	39
7.7 FunTechHouse_RoomTemperature/MQTT_Logic.h File Reference	39
7.7.1 Detailed Description	39
7.8 FunTechHouse_RoomTemperature/Sensor.cpp File Reference	40
7.8.1 Detailed Description	40
7.9 FunTechHouse_RoomTemperature/Sensor.h File Reference	41
7.9.1 Detailed Description	42
7.10 FunTechHouse_RoomTemperature/SensorTypes.h File Reference	42
7.10.1 Detailed Description	42
7.10.2 Enumeration Type Documentation	42
7.10.2.1 FT_SensorType	42
7.11 FunTechHouse_RoomTemperature/StringHelp.cpp File Reference	43
7.11.1 Detailed Description	43
7.12 FunTechHouse_RoomTemperature/StringHelp.h File Reference	44
7.12.1 Detailed Description	44
7.13 FunTechHouse_RoomTemperature/TemperatureSensor.cpp File Reference	45
7.13.1 Detailed Description	45
7.14 FunTechHouse_RoomTemperature/TemperatureSensor.h File Reference	46
7.14.1 Detailed Description	47
7.14.2 Enumeration Type Documentation	47
7.14.2.1 AlarmStates	47
7.14.2.2 SensorAlarmNumber	47
7.15 FunTechHouse_RoomTemperature/ValueAvgInt.cpp File Reference	47
7.15.1 Detailed Description	48
7.16 FunTechHouse_RoomTemperature/ValueAvgInt.h File Reference	48
7.16.1 Detailed Description	48

**Index****50**



## 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](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:

- DS18B20 . . . . . 11
- LVTS . . . . . 12
- MQTT\_Logic . . . . . 14
  - Sensor . . . . . 18
- OneWire . . . . . 17
- PubSubClient . . . . . 17
- StringHelp . . . . . 20
- TemperatureSensor . . . . . 21
  - Sensor . . . . . 18
- ValueAvgInt . . . . . 29



# Chapter 4

## Class Index

### 4.1 Class List

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

- [DS18B20](#)
  - [OneWire DS18B20 Temperature Sensor Class](#) . . . . . 11
- [LVTS](#)
  - Low Voltage Temperature [Sensor](#) Class . . . . . 12
- [MQTT\\_Logic](#)
  - The MQTT logic functions that can be inherited . . . . . 14
- [OneWire](#) . . . . . 17
- [PubSubClient](#) . . . . . 17
- [Sensor](#)
  - A temperature sensor class with alarm logic . . . . . 18
- [StringHelp](#)
  - String helper functions . . . . . 20
- [TemperatureSensor](#)
  - A Temperature sensor class for the [DS18B20](#) and [LVTS](#) . . . . . 21
- [ValueAvgInt](#)
  - A basic filter . . . . . 29



# Chapter 5

## File Index

### 5.1 File List

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

FunTechHouse_RoomTemperature/DS18B20.cpp	
<a href="#">OneWire DS18B20 Temperature Sensor Class</a> . . . . .	33
FunTechHouse_RoomTemperature/DS18B20.h	
<a href="#">OneWire DS18B20 Temperature Sensor Class</a> . . . . .	34
FunTechHouse_RoomTemperature/FunTechHouse_RoomTemperature.ino	
Main file . . . . .	35
FunTechHouse_RoomTemperature/LVTS.cpp	
Low Voltage Temperature <a href="#">Sensor Class</a> . . . . .	36
FunTechHouse_RoomTemperature/LVTS.h	
Low Voltage Temperature <a href="#">Sensor Class</a> . . . . .	37
FunTechHouse_RoomTemperature/MQTT_Logic.cpp	
The MQTT logic with topics for subscribe and publish . . . . .	38
FunTechHouse_RoomTemperature/MQTT_Logic.h	
The MQTT logic with topics for subscribe and publish . . . . .	39
FunTechHouse_RoomTemperature/OneWire.h	??
FunTechHouse_RoomTemperature/PubSubClient.h	??
FunTechHouse_RoomTemperature/Sensor.cpp	
A temperature sensor class with alarm logic . . . . .	40
FunTechHouse_RoomTemperature/Sensor.h	
A temperature sensor class with alarm logic . . . . .	41
FunTechHouse_RoomTemperature/SensorTypes.h	
SensorType has the supported list of sensors . . . . .	42
FunTechHouse_RoomTemperature/StringHelp.cpp	
Helper functions . . . . .	43
FunTechHouse_RoomTemperature/StringHelp.h	
String helper functions . . . . .	44
FunTechHouse_RoomTemperature/TemperatureSensor.cpp	
A temperature sensor class with alarm logic . . . . .	45
FunTechHouse_RoomTemperature/TemperatureSensor.h	
A temperature sensor class with alarm logic . . . . .	46
FunTechHouse_RoomTemperature/ValueAvgInt.cpp	
A basic filter . . . . .	47
FunTechHouse_RoomTemperature/ValueAvgInt.h	
A basic filter . . . . .	48





# 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

<code>in</code>	<code>pin</code>	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.

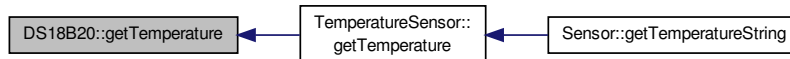
## Parameters

out	<i>value</i>	Temperature reading
-----	--------------	---------------------

## Returns

true if ok, false if fail.

Here is the caller graph for this function:



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	<i>pin</i>	is the IO pin
in	<i>type</i>	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.

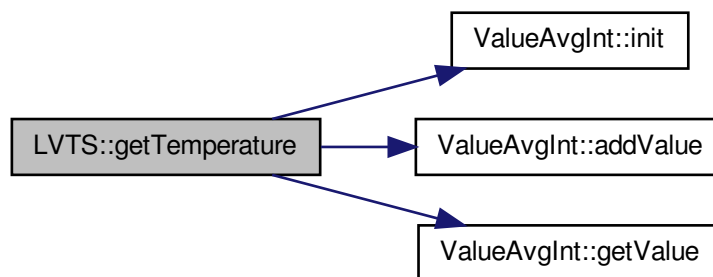
## Parameters

out	<i>value</i>	is the temperature return value
-----	--------------	---------------------------------

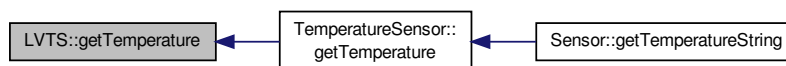
## Returns

true if ok

Here is the call graph for this function:



Here is the caller graph for this function:



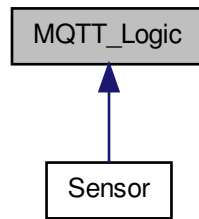
The documentation for this class was generated from the following files:

- [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 constructor.*
- 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

<i>in</i>	<i>check</i>	string to compare with
-----------	--------------	------------------------

##### Returns

true if same, false if not the same.

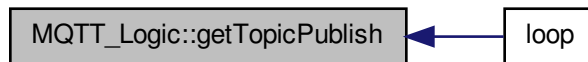
#### 6.3.2.2 char \* MQTT\_Logic::getTopicPublish ( )

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	<i>topicSubscribe</i>	data from the mqtt server
in	<i>topicPublish</i>	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** ()

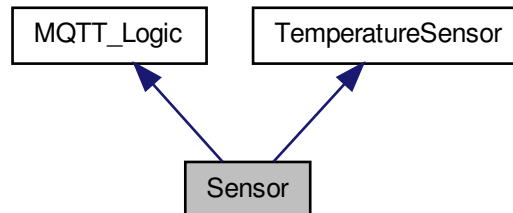
The documentation for this class was generated from the following files:

- FunTechHouse\_RoomTemperature/PubSubClient.h
- FunTechHouse\_RoomTemperature/PubSubClient.cpp

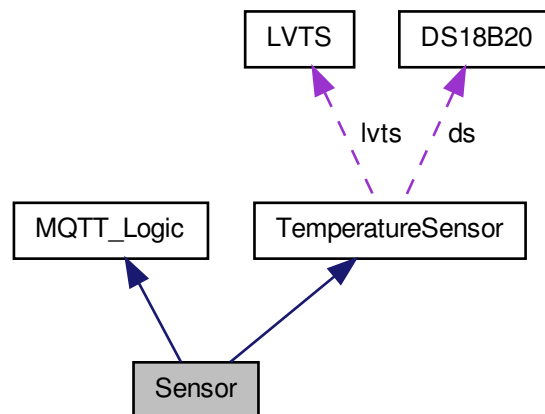
## 6.6 Sensor Class Reference

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.

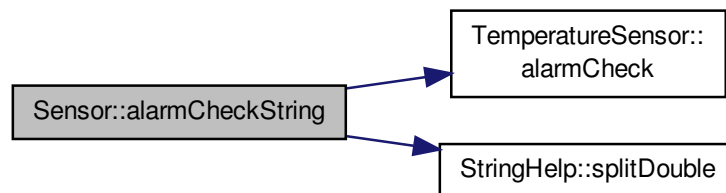
##### Parameters

out	<i>str</i>	returns a string with alarm data
in	<i>size</i>	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.

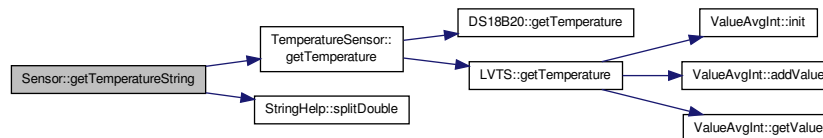
## Parameters

out	<i>str</i>	returns a string with temperature data
in	<i>size</i>	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.

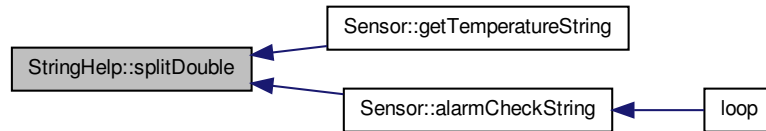
Please note that the decimal part will be 2 digits, so you may need to fill with zero when it is printed. i.e. 4.04 will return 4 and 4, and 5.2 will return 5 and 20.

## Parameters

in	<i>value</i>	The value to split
----	--------------	--------------------

out	<i>integer</i>	The integer part that will be returned
out	<i>decimal</i>	The decimal part that will be returned

Here is the caller graph for this function:



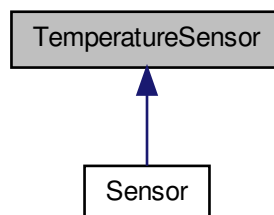
The documentation for this class was generated from the following files:

- FunTechHouse\_RoomTemperature/[StringHelp.h](#)
- FunTechHouse\_RoomTemperature/[StringHelp.cpp](#)

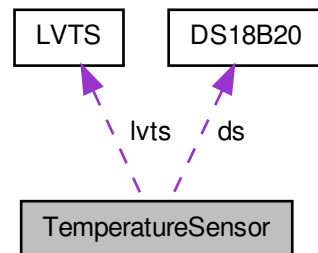
## 6.8 TemperatureSensor Class Reference

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 activateHighAlarm, 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 measured value.*

### 6.8.1 Detailed Description

A Temperature sensor class for the [DS18B20](#) and [LVTS](#).

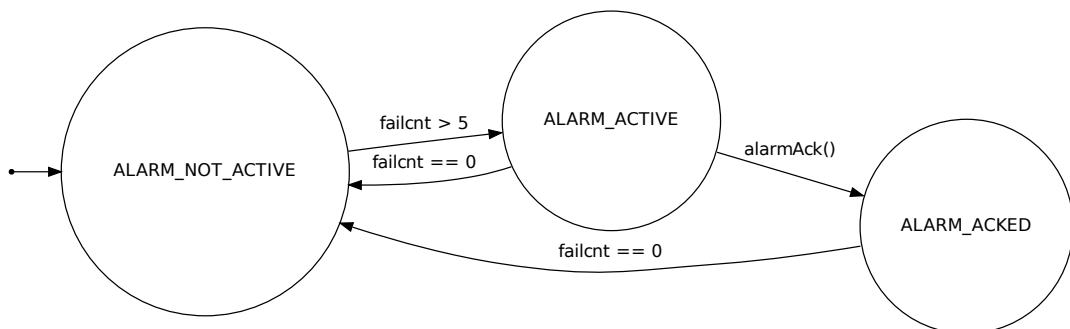
This class will wrap the different sensors and give them the same interface so they can be put in an array and just looped from main.

There is also some alarm logic so that main knows when things are wrong. The alarm is active as long as it is not acknowledged (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](#)

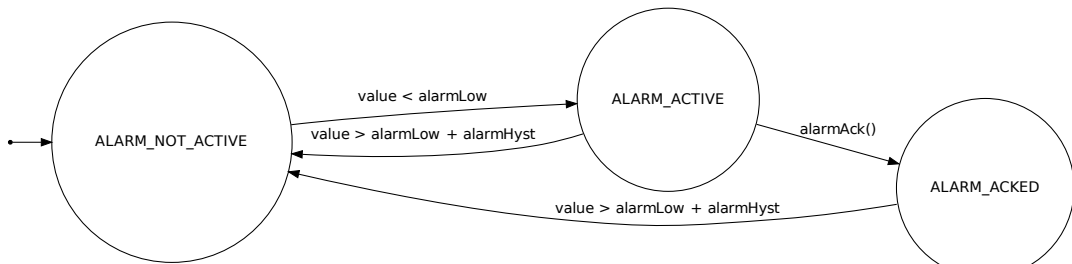


State machine for `SENSOR_ALARM_SENSOR`

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

See Also

[SENSOR\\_ALARM\\_LOW](#)

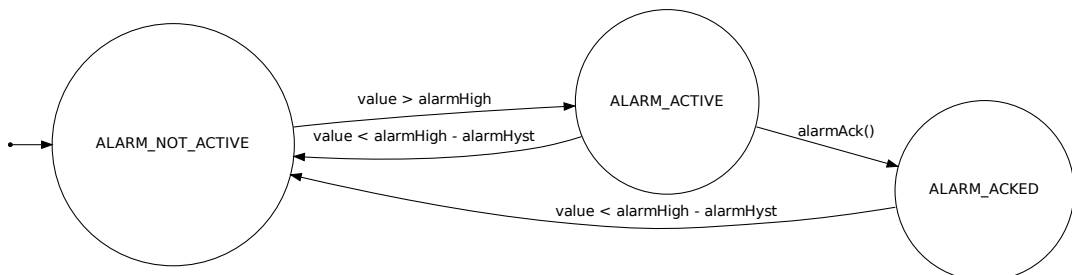


State machine for SENSOR\_ALARM\_LOW

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

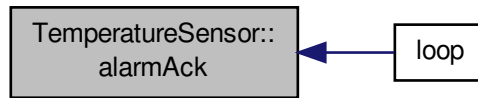
### 6.8.2.1 void TemperatureSensor::alarmAck ( SensorAlarmNumber num )

Acknowledge alarm, dont send any more at this time.

Parameters

<i>num</i>	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.

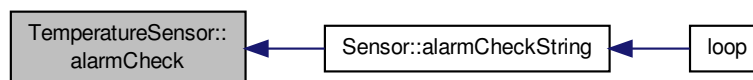
Please note that this should be called after [getTemperature\(\)](#).

\$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.

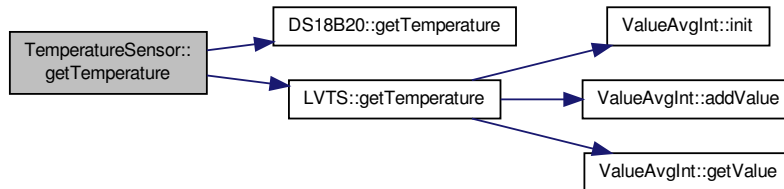
#### Parameters

out	<i>value</i>	is the temperature return value
-----	--------------	---------------------------------

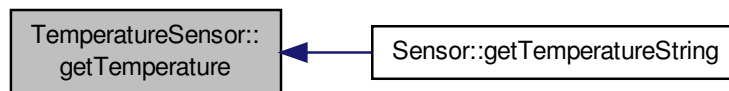
**Returns**

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	<i>pin</i>	is IO pin
in	<i>type</i>	is sensor type i.e. <a href="#">DS18B20</a> 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* )

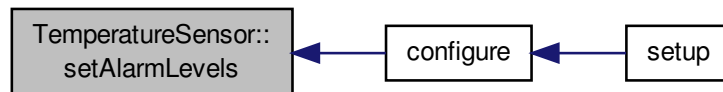
Active alarm and set what alarm levels to be used.

Please note that the low alarm is activated under  $\text{alarmLevelLow} - \text{alarmHyst}$ , and deactivates over  $\text{alarmLevelLow}$ . The high alarm activates over  $\text{alarmLevelHigh} + \text{alarmHyst}$ , and deactivates lower than  $\text{alarmLevelLow}$ .

#### Parameters

<i>alarmHyst</i>	How big hysteresis around the alarm level.
<i>activateLowAlarm</i>	true to activate low alarm.
<i>alarmLevelLow</i>	alarm level for low.
<i>activateHighAlarm</i>	true to active high alarm.
<i>alarmLevelHigh</i>	alarm level for high.

Here is the caller graph for this function:



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

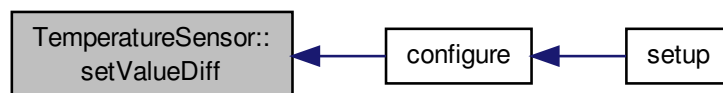
Enable value diff to send value.

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

#### Parameters

<i>diff</i>	the value
-------------	-----------

Here is the caller graph for this function:



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

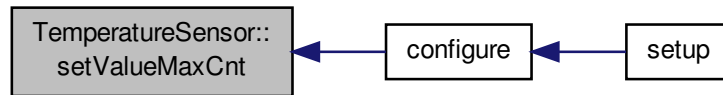
Enable send after X counts even if value is the same.

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

## Parameters

<i>cnt</i>	how many times can we call <code>getTemperature</code> 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

<i>offset</i>	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 collected 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

<code>in</code>	<code>data</code>	is some data to be used int the filter
-----------------	-------------------	--

Here is the caller graph for this function:



### 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:



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

- [FunTechHouse\\_RoomTemperature/ValueAvgInt.h](#)
- [FunTechHouse\\_RoomTemperature/ValueAvgInt.cpp](#)



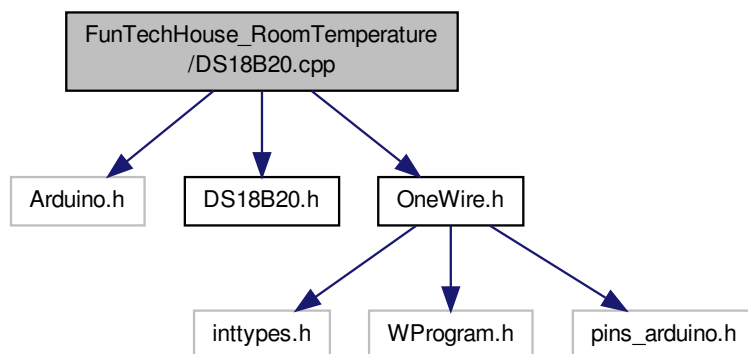
## Chapter 7

# 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.

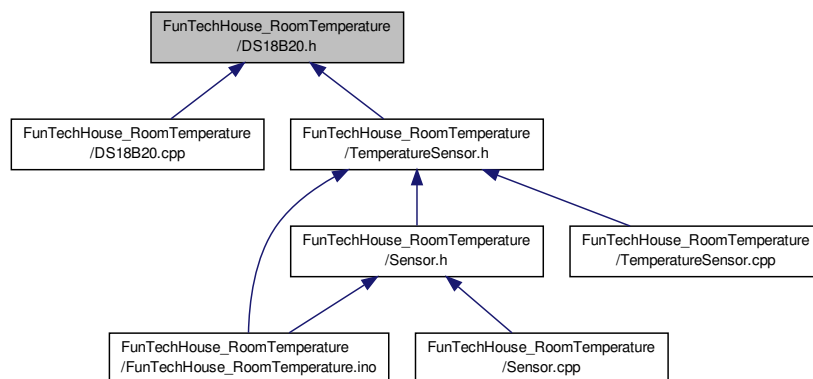
## Author

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:



### Classes

- class [DS18B20](#)

*OneWire DS18B20 Temperature Sensor Class.*

### 7.2.1 Detailed Description

[OneWire DS18B20 Temperature Sensor Class](#).



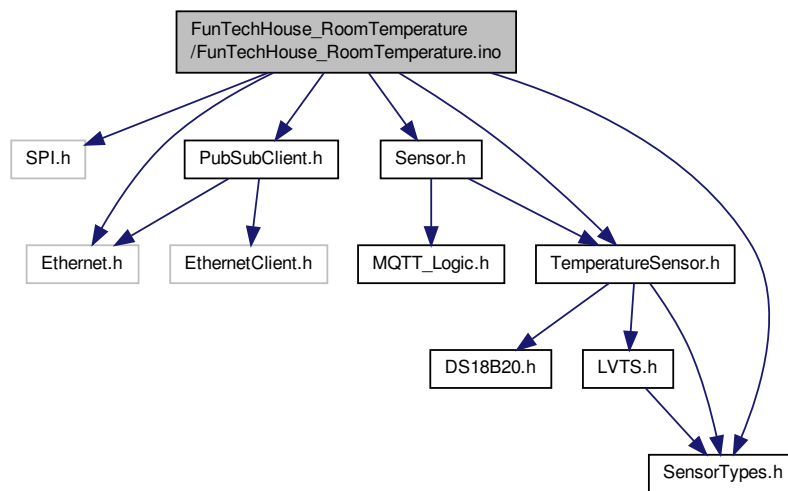
## Author

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.

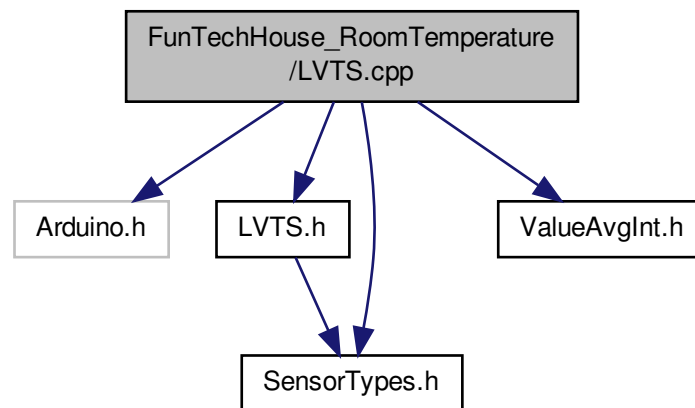
#### Parameters

<i>in</i>	<i>topic</i>	What mqtt topic triggered this callback
<i>in</i>	<i>payload</i>	The actual message
<i>in</i>	<i>length</i>	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.

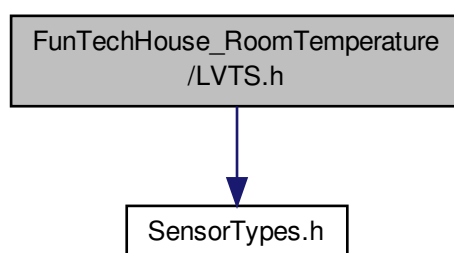
Author

Johan Simonsson

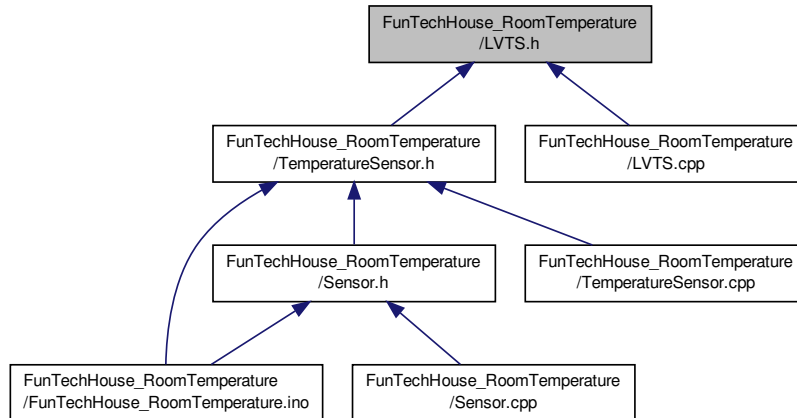
## 7.5 FunTechHouse\_RoomTemperature/LVTS.h File Reference

Low Voltage Temperature [Sensor](#) Class.

Include dependency graph for LVTS.h:



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



## Classes

- class [LVTS](#)

*Low Voltage Temperature [Sensor](#) Class.*

### 7.5.1 Detailed Description

Low Voltage Temperature [Sensor](#) Class.

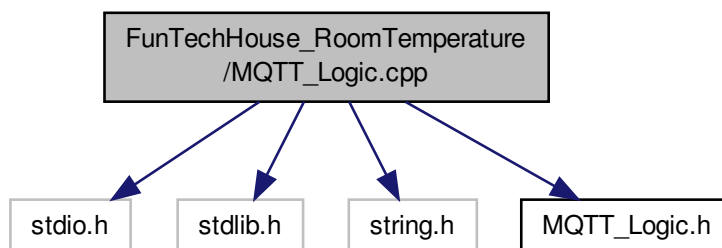
#### Author

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.

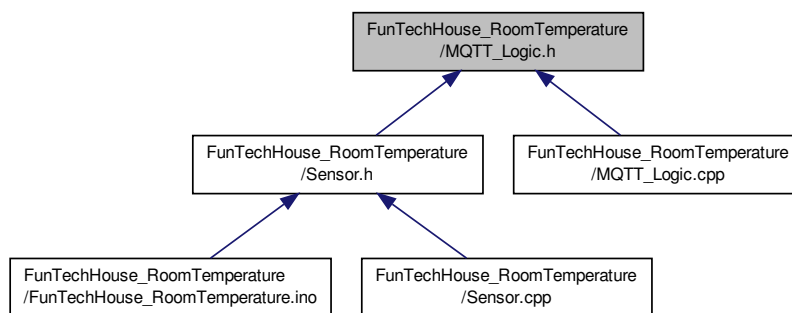
#### Author

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:



### 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.

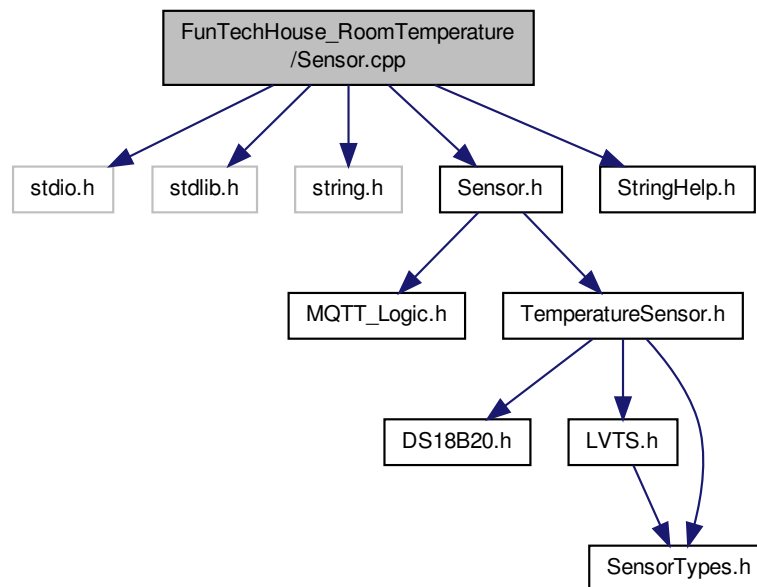
## Author

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.

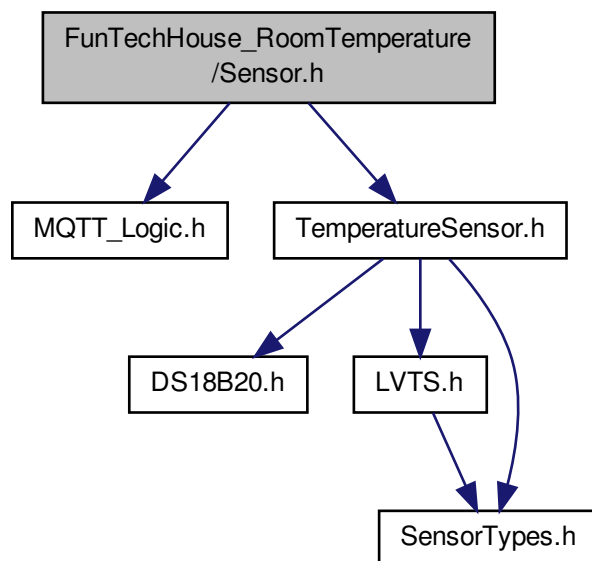
Author

Johan Simonsson

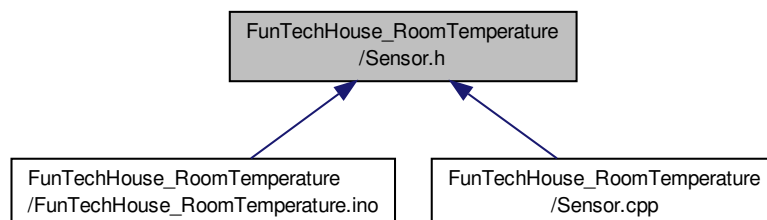
## 7.9 FunTechHouse\_RoomTemperature/Sensor.h File Reference

A temperature sensor class with alarm logic.

Include dependency graph for Sensor.h:



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



### Classes

- class [Sensor](#)

*A temperature sensor class with alarm logic.*

### 7.9.1 Detailed Description

A temperature sensor class with alarm logic.

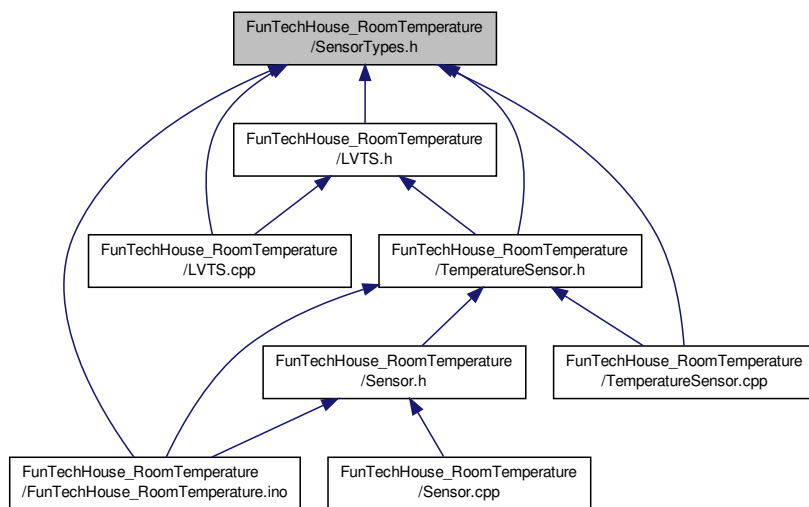
#### Author

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:



### Enumerations

- enum `FT_SensorType` { `SENSOR_NONE` = 0, `SENSOR_DS18B20`, `SENSOR_LVTS_LM34`, `SENSOR_LVTS_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.

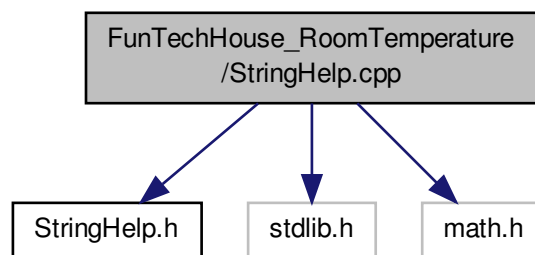
***SENSOR\_LVTS\_LM34*** LM34 a low voltage temperature sensor. 10mV per degF.

***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.

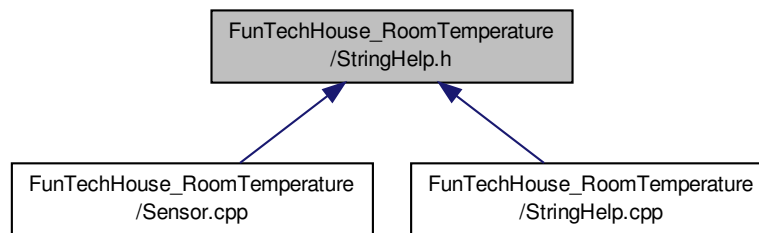
**Author**

Johan Simonsson

**7.12 FunTechHouse\_RoomTemperature/StringHelp.h File Reference**

String helper functions.

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

**Classes**

- class [StringHelp](#)

*String helper functions.*

**7.12.1 Detailed Description**

String helper functions.

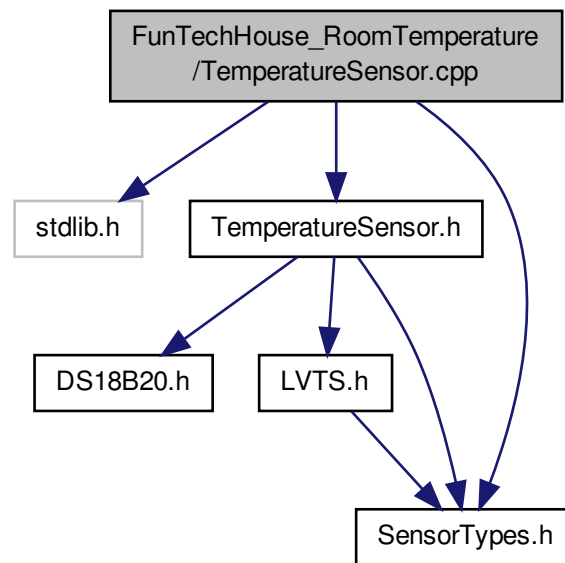
Author

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.

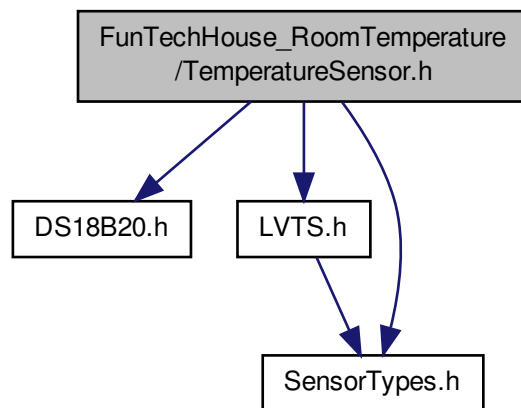
## Author

Johan Simonsson

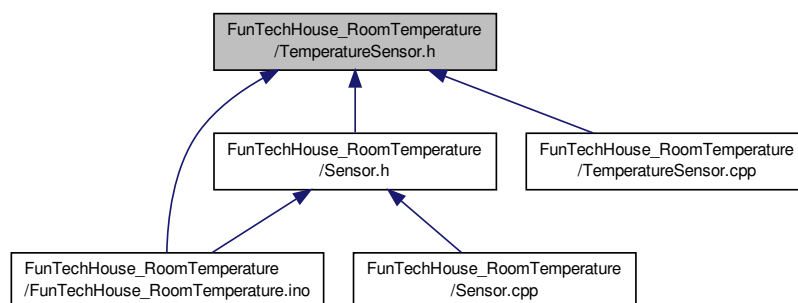
## 7.14 FunTechHouse\_RoomTemperature/TemperatureSensor.h File Reference

A temperature sensor class with alarm logic.

Include dependency graph for TemperatureSensor.h:



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



### 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.

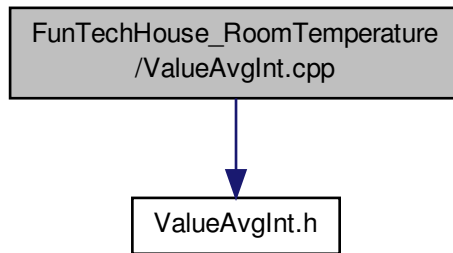
**`SENSOR_ALARM_HIGH`** High level alarm.

**`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.

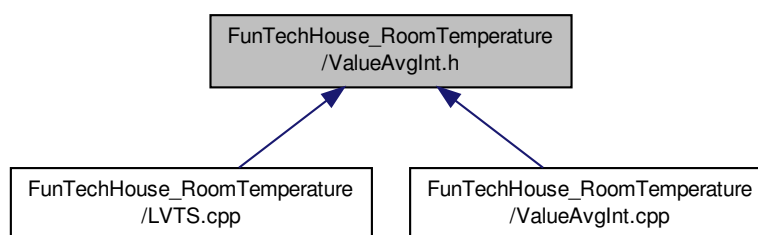
Author

Johan Simonsson

## 7.16 FunTechHouse\_RoomTemperature/ValueAvgInt.h File Reference

A basic filter.

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



### Classes

- class [ValueAvgInt](#)  
*A basic filter.*

### 7.16.1 Detailed Description

A basic filter.

Author

Johan Simonsson

# Index

- ALARM\_ACKED
    - TemperatureSensor.h, 47
  - ALARM\_ACTIVE
    - TemperatureSensor.h, 47
  - ALARM\_NOT\_ACTIVE
    - TemperatureSensor.h, 47
  - addValue
    - ValueAvgInt, 30
  - alarmAck
    - TemperatureSensor, 24
  - alarmCheck
    - TemperatureSensor, 25
  - alarmCheckString
    - Sensor, 19
  - AlarmStates
    - TemperatureSensor.h, 47
  
  - callback
    - FunTechHouse\_RoomTemperature.ino, 36
  - checkTopicSubscribe
    - MQTT\_Logic, 15
  
  - DS18B20, 11
    - DS18B20, 11
    - DS18B20, 11
    - getTemperature, 11
  
  - FT\_SensorType
    - SensorTypes.h, 42
  - FunTechHouse\_RoomTemperature.ino
    - callback, 36
  - FunTechHouse\_RoomTemperature/DS18B20.cpp, 33
  - FunTechHouse\_RoomTemperature/DS18B20.h, 34
  - FunTechHouse\_RoomTemperature/FunTechHouse\_RoomTemperature.ino, 35
  - FunTechHouse\_RoomTemperature/LVTS.cpp, 36
  - FunTechHouse\_RoomTemperature/LVTS.h, 37
  - FunTechHouse\_RoomTemperature/MQTT\_Logic.cpp, 38
  - FunTechHouse\_RoomTemperature/MQTT\_Logic.h, 39
  - FunTechHouse\_RoomTemperature/Sensor.cpp, 40
  - FunTechHouse\_RoomTemperature/Sensor.h, 41
  - FunTechHouse\_RoomTemperature/SensorTypes.h, 42
  - FunTechHouse\_RoomTemperature/StringHelp.cpp, 43
  - FunTechHouse\_RoomTemperature/StringHelp.h, 44
  - FunTechHouse\_RoomTemperature/Temperature-Sensor.cpp, 45
  - FunTechHouse\_RoomTemperature/Temperature-Sensor.h, 46
  - FunTechHouse\_RoomTemperature/ValueAvgInt.cpp, 47
- FunTechHouse\_RoomTemperature/ValueAvgInt.h, 48
- getTemperature
    - DS18B20, 11
    - LVTS, 13
    - TemperatureSensor, 25
  - getTemperatureString
    - Sensor, 19
  - getTopicPublish
    - MQTT\_Logic, 15
  - getTopicSubscribe
    - MQTT\_Logic, 16
  - getValue
    - ValueAvgInt, 31
  
  - init
    - TemperatureSensor, 26
  
  - LVTS, 12
    - getTemperature, 13
    - LVTS, 12
    - LVTS, 12
  
  - MQTT\_Logic, 14
    - checkTopicSubscribe, 15
    - getTopicPublish, 15
    - getTopicSubscribe, 16
    - setTopic, 16
  
  - OneWire, 17
  
  - PubSubClient, 17
  
  - SENSOR\_ALARM\_HIGH
    - TemperatureSensor.h, 47
  - SENSOR\_ALARM\_LOW
    - TemperatureSensor.h, 47
  - SENSOR\_ALARM\_NO
    - TemperatureSensor.h, 47
  - SENSOR\_ALARM\_SENSOR
    - TemperatureSensor.h, 47
  - SENSOR\_DS18B20
    - SensorTypes.h, 43
  - SENSOR\_LVTS\_LM34
    - SensorTypes.h, 43
  - SENSOR\_LVTS\_LM35
    - SensorTypes.h, 43
  - SENSOR\_NONE
    - SensorTypes.h, 43
  - Sensor, 18
    - alarmCheckString, 19



- getTemperatureString, [19](#)
- SensorTypes.h
  - SENSOR\_DS18B20, [43](#)
  - SENSOR\_LVTS\_LM34, [43](#)
  - SENSOR\_LVTS\_LM35, [43](#)
  - SENSOR\_NONE, [43](#)
- SensorAlarmNumber
  - TemperatureSensor.h, [47](#)
- SensorTypes.h
  - FT\_SensorType, [42](#)
- setAlarmLevels
  - TemperatureSensor, [26](#)
- setTopic
  - MQTT\_Logic, [16](#)
- setValueDiff
  - TemperatureSensor, [27](#)
- setValueMaxCnt
  - TemperatureSensor, [27](#)
- setValueOffset
  - TemperatureSensor, [29](#)
- splitDouble
  - StringHelp, [20](#)
- StringHelp, [20](#)
  - splitDouble, [20](#)
- TemperatureSensor.h
  - ALARM\_ACKED, [47](#)
  - ALARM\_ACTIVE, [47](#)
  - ALARM\_NOT\_ACTIVE, [47](#)
  - SENSOR\_ALARM\_HIGH, [47](#)
  - SENSOR\_ALARM\_LOW, [47](#)
  - SENSOR\_ALARM\_NO, [47](#)
  - SENSOR\_ALARM\_SENSOR, [47](#)
- TemperatureSensor, [21](#)
  - alarmAck, [24](#)
  - alarmCheck, [25](#)
  - getTemperature, [25](#)
  - init, [26](#)
  - setAlarmLevels, [26](#)
  - setValueDiff, [27](#)
  - setValueMaxCnt, [27](#)
  - setValueOffset, [29](#)
- TemperatureSensor.h
  - AlarmStates, [47](#)
  - SensorAlarmNumber, [47](#)
- ValueAvgInt, [29](#)
  - addValue, [30](#)
  - getValue, [31](#)