

## AM2302-Sensor

Generated on Sat Mar 28 2026 23:00:07 for AM2302-Sensor by Doxygen 1.9.8

Sat Mar 28 2026 23:00:07



<b>1 AM2302-Sensor Library</b>	<b>1</b>
1.1 Contents	1
1.2 Sensor Documentation	1
1.3 Library Usage	2
1.3.1 Controllers	2
1.3.2 Usage the AM2302-Sensor library in the Code	2
1.3.3 Status Codes of AM2302-Sensor	2
1.4 Unit Tests	2
1.4.1 Run tests locally	3
1.4.1.1 Windows	3
1.4.1.2 Linux	3
1.4.1.3 macOS	3
1.4.2 Continuous Integration	3
1.5 License	3
1.6 Helpful Links	3
<b>2 Class Index</b>	<b>5</b>
2.1 Class List	5
<b>3 File Index</b>	<b>7</b>
3.1 File List	7
<b>4 Class Documentation</b>	<b>9</b>
4.1 AM2302::AM2302_Sensor Class Reference	9
4.1.1 Constructor & Destructor Documentation	9
4.1.1.1 AM2302_Sensor()	9
4.1.2 Member Function Documentation	10
4.1.2.1 begin()	10
4.1.2.2 get_sensorState()	10
4.1.2.3 read()	10
<b>5 File Documentation</b>	<b>11</b>
5.1 src/AM2302-Sensor.cpp File Reference	11
5.1.1 Detailed Description	11
5.2 src/AM2302-Sensor.h File Reference	12
5.2.1 Detailed Description	13
5.3 AM2302-Sensor.h	13
<b>Index</b>	<b>15</b>



# Chapter 1

## AM2302-Sensor Library

Sensor Library for the AM2302 Sensor (aka DHT22) from ASAIR.

This Library is a controller independent library for reading the AM2302 Sensor also known as DHT22.

### 1.1 Contents

- Sensor Documentation
- AM2302 Library Usage
- Unit Tests
- License
- Helpful Links

### 1.2 Sensor Documentation

One small Docu you will find in the docs folder of this repo.

The actual manufacturer page is linked here: [ASAIR AM2302](#).

The most detailed datasheet you will find here: [AM2302 Product Manual](#).

The Sensor-Library documentation you will find in the provided [refman.pdf](#), or under [github-pages github.io](#).

#### REMARK:

Against the most documentations and datasheets the following Pin description is correct (from left to right):

- Pin1: VDD (3,3...5 V)
- Pin2: SDA (Serial Data, two way)
- Pin3: GND
- PIN4: GND

## 1.3 Library Usage

### 1.3.1 Controllers

The library is intended to be used on each microcontroller for Example:

- Arduino Nano
- Arduino Nano 33 IOT
- ESP8266
- ESP32
- etc ...

### 1.3.2 Usage the AM2302-Sensor library in the Code

Include the library

```
#include <AM2302-Sensor.h>
```

The library use namespaces, so the object can be instantiated and used by:

```
AM2302::AM2302_Sensor am2302{PIN};

void setup() {
    am2302.begin();

    auto status = am2302.read();
    Serial.print("\n\nstatus of sensor read(): ");
    Serial.println(AM2302::AM2302_Sensor::get_sensorState(status));

    Serial.print("Temperature: ");
    Serial.println(am2302.get_Temperature());

    Serial.print("Humidity: ");
    Serial.println(am2302.get_Humidity());
}
```

### 1.3.3 Status Codes of AM2302-Sensor

The following status codes exists:

- AM2302\_READ\_OK {0};
- AM2302\_ERROR\_CHECKSUM {-1};
- AM2302\_ERROR\_TIMEOUT {-2};
- AM2302\_ERROR\_READ\_FREQ {-3};

## 1.4 Unit Tests

The project contains unit tests for the pure helper functions in [src/AM2302-Sensor\\_Tools.h](#). These tests validate:

- checksum calculation
- humidity decoding
- temperature decoding
- regression cases for bit shift and sign handling

### 1.4.1 Run tests locally

The tests are based on GoogleTest and are built with CMake.

Required tools:

- CMake
- a C++17 compatible compiler

#### 1.4.1.1 Windows

With Visual Studio or Visual Studio Build Tools installed, run the commands in a Developer PowerShell:

```
# Generate Build System
cmake -B build -DCMAKE_BUILD_TYPE=Release
# Build a Project
cmake --build build
# Execute Tests
ctest --test-dir build --output-on-failure
```

#### 1.4.1.2 Linux

Install CMake and a compiler toolchain, for example g++ or clang++, and run:

```
# Generate Build System
cmake -B build -DCMAKE_BUILD_TYPE=Release
# Build a Project
cmake --build build
# Execute Tests
ctest --test-dir build --output-on-failure
```

#### 1.4.1.3 macOS

Install Xcode Command Line Tools and CMake, then run:

```
# Generate Build System
cmake -B build -DCMAKE_BUILD_TYPE=Release
# Build a Project
cmake --build build
# Execute Tests
ctest --test-dir build --output-on-failure
```

### 1.4.2 Continuous Integration

The GitHub Actions workflow `unit_tests.yml` builds and runs the unit tests automatically on every push and pull request.

## 1.5 License

This library is licensed under MIT Licence.

[AM2302-Sensor License](#)

## 1.6 Helpful Links

- [ESP8266-01-Adapter](#)
- [AM2302-Adapter](#)





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AM2302::AM2302_Sensor</a> . . . . .	9
---	---



# Chapter 3

## File Index

### 3.1 File List

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

src/ <a href="#">AM2302-Sensor.cpp</a>	
Measure Temperature and Humidity of AM2302-Sensor . . . . .	<a href="#">11</a>
src/ <a href="#">AM2302-Sensor.h</a>	
Measure Temperature and Humidity of AM2302-Sensor . . . . .	<a href="#">12</a>



# Chapter 4

## Class Documentation

### 4.1 AM2302::AM2302\_Sensor Class Reference

#### Public Member Functions

- [AM2302\\_Sensor](#) (uint8\_t pin)  
*Construct a new am2302::am2302 sensor::am2302 sensor object.*
- bool [begin](#) ()  
*begin function, setup pin and run sensor check.*
- int8\_t [read](#) ()  
*read function, call of read\_sensor()*
- float **get\_Temperature** () const
- float **get\_Humidity** () const

#### Static Public Member Functions

- static const char \* [get\\_sensorState](#) (int8\_t state)  
*get Sensor State in human readable manner*

#### 4.1.1 Constructor & Destructor Documentation

##### 4.1.1.1 AM2302\_Sensor()

```
AM2302::AM2302_Sensor::AM2302_Sensor (
    uint8_t pin ) [explicit]
```

Construct a new am2302::am2302 sensor::am2302 sensor object.

#### Parameters

<i>pin</i>	Pin for AM2302 sensor
------------	-----------------------

## 4.1.2 Member Function Documentation

### 4.1.2.1 begin()

```
bool AM2302::AM2302_Sensor::begin ( )
```

begin function, setup pin and run sensor check.

#### Returns

true if sensor check is successful.  
false if sensor check failed.

### 4.1.2.2 get\_sensorState()

```
const char * AM2302::AM2302_Sensor::get_sensorState (
    int8_t state ) [static]
```

get Sensor State in human readable manner

#### Returns

sensor state : OK, Checksum Error or Timeout Error

### 4.1.2.3 read()

```
int8_t AM2302::AM2302_Sensor::read ( )
```

read function, call of read\_sensor()

#### Returns

sensor status

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

- [src/AM2302-Sensor.h](#)
- [src/AM2302-Sensor.cpp](#)

## Chapter 5

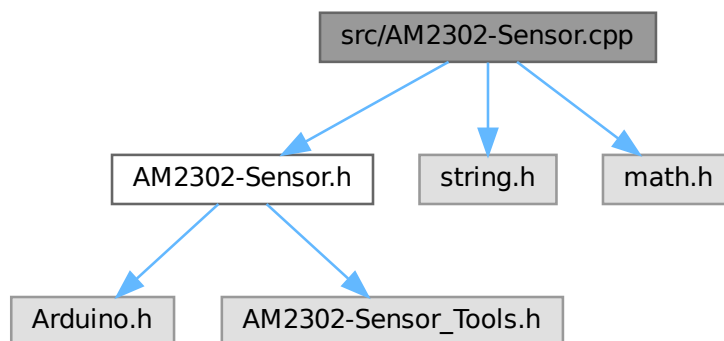
# File Documentation

### 5.1 src/AM2302-Sensor.cpp File Reference

Measure Temperature and Humidity of AM2302-Sensor.

```
#include <AM2302-Sensor.h>
#include <string.h>
#include <math.h>
```

Include dependency graph for AM2302-Sensor.cpp:



#### 5.1.1 Detailed Description

Measure Temperature and Humidity of AM2302-Sensor.

Author

Frank Häfele

Date

23.03.2026

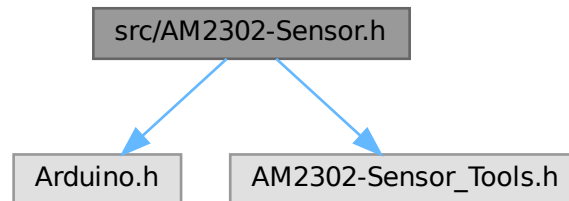
Version

1.5.0

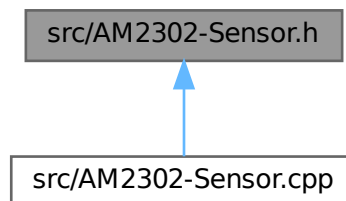
## 5.2 src/AM2302-Sensor.h File Reference

Measure Temperature and Humidity of AM2302-Sensor.

```
#include <Arduino.h>
#include "AM2302-Sensor_Tools.h"
Include dependency graph for AM2302-Sensor.h:
```



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



### Classes

- class [AM2302::AM2302\\_Sensor](#)

### Variables

- constexpr const char \* **AM2302::AM2302\_STATE\_OK** {"OK"}
- constexpr const char \* **AM2302::AM2302\_STATE\_ERR\_CKSUM** {"Error: Checksum"}
- constexpr const char \* **AM2302::AM2302\_STATE\_ERR\_TIMEOUT** {"Error: Timeout"}
- constexpr const char \* **AM2302::AM2302\_STATE\_ERR\_READ\_FREQ** {"Error: Read Frequency"}
- constexpr int8\_t **AM2302::AM2302\_READ\_OK** {0}
- constexpr int8\_t **AM2302::AM2302\_ERROR\_CHECKSUM** {-1}
- constexpr int8\_t **AM2302::AM2302\_ERROR\_TIMEOUT** {-2}
- constexpr int8\_t **AM2302::AM2302\_ERROR\_READ\_FREQ** {-3}
- constexpr uint8\_t **AM2302::READ\_TIMEOUT** {100U}
- constexpr uint16\_t **AM2302::READ\_FREQUENCY** {2000U}



## 5.2.1 Detailed Description

Measure Temperature and Humidity of AM2302-Sensor.

### Author

Frank Häfele

### Date

23.03.2026

### Version

1.5.0

## 5.3 AM2302-Sensor.h

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

```

00001
00010 #ifndef __AM2302_SENSOR_H__
00011 #define __AM2302_SENSOR_H__
00012
00013 #include <Arduino.h>
00014 #include "AM2302-Sensor_Tools.h"
00015
00016 namespace AM2302 {
00017
00018     constexpr const char * AM2302_STATE_OK{"OK"};
00019     constexpr const char * AM2302_STATE_ERR_CKSUM{"Error: Checksum"};
00020     constexpr const char * AM2302_STATE_ERR_TIMEOUT{"Error: Timeout"};
00021     constexpr const char * AM2302_STATE_ERR_READ_FREQ{"Error: Read Frequency"};
00022
00023     constexpr int8_t AM2302_READ_OK {0};
00024     constexpr int8_t AM2302_ERROR_CHECKSUM {-1};
00025     constexpr int8_t AM2302_ERROR_TIMEOUT {-2};
00026     constexpr int8_t AM2302_ERROR_READ_FREQ {-3};
00027
00028     // define timeout in 100 µs
00029     constexpr uint8_t READ_TIMEOUT {100U};
00030
00031     // define maximum sensor read frequency in milliseconds (2 s)
00032     constexpr uint16_t READ_FREQUENCY {2000U};
00033
00034     class AM2302_Sensor {
00035     public:
00036         explicit AM2302_Sensor(uint8_t pin);
00042
00043         bool begin();
00050
00051         int8_t read();
00057
00058
00059         inline float get_Temperature() const {return _temp;}
00060         inline float get_Humidity() const {return _hum;}
00061
00062         static const char * get_sensorState(int8_t state);
00068
00069     private:
00070         // memory millis between last read
00071         unsigned long _millis_last_read;
00072         // memory for sensor data
00073         uint8_t _data[5] = {0};
00074         // holds humidity
00075         float _hum;
00076         // holds temperature
00077         float _temp;
00078         // sensor pin
00079         uint8_t _pin;
00080         // holds checksum state
00081

```

```
00082     bool _checksum_ok {false};
00083
00090     int8_t await_pin_state(uint8_t state);
00091
00097     int8_t read_sensor();
00098
00106     int8_t read_sensor_data(uint8_t *buffer, uint8_t const size);
00107
00112     void resetData();
00113 };
00114 }
00115
00116 namespace AM2302_Tools {
00122     void print_byte_as_bit(const uint8_t value);
00123 }
00124
00125 #endif
```

# Index

- AM2302-Sensor Library, [1](#)
- AM2302::AM2302\_Sensor, [9](#)
  - AM2302\_Sensor, [9](#)
  - begin, [10](#)
  - get\_sensorState, [10](#)
  - read, [10](#)
- AM2302\_Sensor
  - AM2302::AM2302\_Sensor, [9](#)
- begin
  - AM2302::AM2302\_Sensor, [10](#)
- get\_sensorState
  - AM2302::AM2302\_Sensor, [10](#)
- read
  - AM2302::AM2302\_Sensor, [10](#)
- src/AM2302-Sensor.cpp, [11](#)
- src/AM2302-Sensor.h, [12](#), [13](#)