

Bus Bridge Server

commit-0c1a9b4

Generated by Doxygen 1.9.8

| | |
|--|-----------|
| 1 Bus Bridge Server | 1 |
| 2 Test List | 3 |
| 3 Hierarchical Index | 5 |
| 3.1 Class Hierarchy | 5 |
| 4 Class Index | 7 |
| 4.1 Class List | 7 |
| 5 File Index | 9 |
| 5.1 File List | 9 |
| 6 Class Documentation | 11 |
| 6.1 BaseSlave Class Reference | 11 |
| 6.1.1 Detailed Description | 12 |
| 6.1.2 Constructor & Destructor Documentation | 12 |
| 6.1.2.1 ~BaseSlave() | 12 |
| 6.1.3 Member Function Documentation | 12 |
| 6.1.3.1 getData() | 12 |
| 6.1.3.2 getId() | 13 |
| 6.1.3.3 getStatus() | 13 |
| 6.1.3.4 setData() | 13 |
| 6.1.3.5 start() | 13 |
| 6.1.3.6 stop() | 13 |
| 6.2 BusServer Class Reference | 14 |
| 6.2.1 Detailed Description | 14 |
| 6.2.2 Constructor & Destructor Documentation | 15 |
| 6.2.2.1 BusServer() | 15 |
| 6.2.3 Member Function Documentation | 15 |
| 6.2.3.1 getSlaveManager() | 15 |
| 6.2.3.2 listen() | 15 |
| 6.2.3.3 send() | 15 |
| 6.2.3.4 setup() | 16 |
| 6.2.3.5 start() | 16 |
| 6.2.4 Member Data Documentation | 16 |
| 6.2.4.1 buffer | 16 |
| 6.2.4.2 listening_address | 17 |
| 6.2.4.3 listening_fd | 17 |
| 6.2.4.4 slave_manager | 17 |
| 6.2.4.5 wemos_bridge_connected | 17 |
| 6.3 CO2Slave Class Reference | 17 |
| 6.3.1 Detailed Description | 19 |
| 6.3.2 Constructor & Destructor Documentation | 20 |

| | |
|--|----|
| 6.3.2.1 CO2Slave() | 20 |
| 6.3.3 Member Function Documentation | 20 |
| 6.3.3.1 getData() | 20 |
| 6.3.3.2 getId() | 20 |
| 6.3.3.3 getStatus() | 20 |
| 6.3.3.4 setData() | 20 |
| 6.3.3.5 start() | 21 |
| 6.3.3.6 stop() | 21 |
| 6.3.4 Member Data Documentation | 21 |
| 6.3.4.1 command | 21 |
| 6.3.4.2 fd | 21 |
| 6.3.4.3 i2c_address | 21 |
| 6.3.4.4 id | 21 |
| 6.3.4.5 power_state | 21 |
| 6.3.4.6 state_packet | 22 |
| 6.4 RGBData Struct Reference | 22 |
| 6.4.1 Detailed Description | 22 |
| 6.4.2 Member Data Documentation | 23 |
| 6.4.2.1 B | 23 |
| 6.4.2.2 G | 23 |
| 6.4.2.3 R | 23 |
| 6.5 RGBSlave Class Reference | 23 |
| 6.5.1 Detailed Description | 25 |
| 6.5.2 Constructor & Destructor Documentation | 26 |
| 6.5.2.1 RGBSlave() | 26 |
| 6.5.3 Member Function Documentation | 26 |
| 6.5.3.1 getData() | 26 |
| 6.5.3.2 getId() | 26 |
| 6.5.3.3 getStatus() | 26 |
| 6.5.3.4 setData() | 26 |
| 6.5.3.5 start() | 27 |
| 6.5.3.6 stop() | 27 |
| 6.5.4 Member Data Documentation | 27 |
| 6.5.4.1 color_state | 27 |
| 6.5.4.2 fd | 27 |
| 6.5.4.3 i2c_address | 27 |
| 6.5.4.4 id | 27 |
| 6.5.4.5 state_packet | 28 |
| 6.6 sensor_data Union Reference | 28 |
| 6.6.1 Detailed Description | 28 |
| 6.6.2 Member Data Documentation | 29 |
| 6.6.2.1 co2 | 29 |

| | |
|--|----|
| 6.6.2.2 generic | 29 |
| 6.6.2.3 heartbeat | 29 |
| 6.6.2.4 humidity | 29 |
| 6.6.2.5 light | 29 |
| 6.6.2.6 rgb_light | 29 |
| 6.6.2.7 temperature | 29 |
| 6.7 sensor_packet::sensor_data Union Reference | 30 |
| 6.7.1 Detailed Description | 30 |
| 6.7.2 Member Data Documentation | 30 |
| 6.7.2.1 co2 | 30 |
| 6.7.2.2 generic | 30 |
| 6.7.2.3 heartbeat | 31 |
| 6.7.2.4 humidity | 31 |
| 6.7.2.5 light | 31 |
| 6.7.2.6 rgb_light | 31 |
| 6.7.2.7 temperature | 31 |
| 6.8 sensor_header Struct Reference | 32 |
| 6.8.1 Detailed Description | 32 |
| 6.8.2 Member Data Documentation | 32 |
| 6.8.2.1 length | 32 |
| 6.8.2.2 ptype | 33 |
| 6.9 sensor_heartbeat Struct Reference | 33 |
| 6.9.1 Detailed Description | 34 |
| 6.9.2 Member Data Documentation | 34 |
| 6.9.2.1 metadata | 34 |
| 6.10 sensor_metadata Struct Reference | 34 |
| 6.10.1 Detailed Description | 35 |
| 6.10.2 Member Data Documentation | 35 |
| 6.10.2.1 sensor_id | 35 |
| 6.10.2.2 sensor_type | 35 |
| 6.11 sensor_packet Struct Reference | 35 |
| 6.11.1 Detailed Description | 36 |
| 6.11.2 Member Data Documentation | 37 |
| 6.11.2.1 data | 37 |
| 6.11.2.2 header | 37 |
| 6.12 sensor_packet_co2 Struct Reference | 37 |
| 6.12.1 Detailed Description | 38 |
| 6.12.2 Member Data Documentation | 38 |
| 6.12.2.1 metadata | 38 |
| 6.12.2.2 value | 38 |
| 6.13 sensor_packet_generic Struct Reference | 39 |
| 6.13.1 Detailed Description | 39 |

| | |
|---|----|
| 6.13.2 Member Data Documentation | 40 |
| 6.13.2.1 metadata | 40 |
| 6.14 sensor_packet_humidity Struct Reference | 40 |
| 6.14.1 Detailed Description | 41 |
| 6.14.2 Member Data Documentation | 41 |
| 6.14.2.1 metadata | 41 |
| 6.14.2.2 value | 41 |
| 6.15 sensor_packet_light Struct Reference | 41 |
| 6.15.1 Detailed Description | 42 |
| 6.15.2 Member Data Documentation | 42 |
| 6.15.2.1 metadata | 42 |
| 6.15.2.2 target_state | 43 |
| 6.16 sensor_packet_rgb_light Struct Reference | 43 |
| 6.16.1 Detailed Description | 44 |
| 6.16.2 Member Data Documentation | 44 |
| 6.16.2.1 blue_state | 44 |
| 6.16.2.2 green_state | 44 |
| 6.16.2.3 metadata | 44 |
| 6.16.2.4 red_state | 44 |
| 6.17 sensor_packet_temperature Struct Reference | 45 |
| 6.17.1 Detailed Description | 45 |
| 6.17.2 Member Data Documentation | 46 |
| 6.17.2.1 metadata | 46 |
| 6.17.2.2 value | 46 |
| 6.18 SlaveManager Class Reference | 46 |
| 6.18.1 Detailed Description | 47 |
| 6.18.2 Constructor & Destructor Documentation | 47 |
| 6.18.2.1 SlaveManager() | 47 |
| 6.18.3 Member Function Documentation | 47 |
| 6.18.3.1 createSlave() | 47 |
| 6.18.3.2 deleteSlave() | 47 |
| 6.18.3.3 getSlave() | 48 |
| 6.18.3.4 initialize() | 48 |
| 6.18.3.5 ledControl() | 48 |
| 6.18.4 Member Data Documentation | 48 |
| 6.18.4.1 address | 48 |
| 6.18.4.2 i2c_fd | 49 |
| 6.18.4.3 initialized | 49 |
| 6.18.4.4 slaves | 49 |
| 6.19 TemperatureData Struct Reference | 49 |
| 6.19.1 Detailed Description | 50 |
| 6.19.2 Member Data Documentation | 50 |

| | |
|---|-----------|
| 6.19.2.1 Temp | 50 |
| 6.20 TemperatureSlave Class Reference | 50 |
| 6.20.1 Detailed Description | 53 |
| 6.20.2 Constructor & Destructor Documentation | 53 |
| 6.20.2.1 TemperatureSlave() | 53 |
| 6.20.3 Member Function Documentation | 53 |
| 6.20.3.1 getData() | 53 |
| 6.20.3.2 getId() | 53 |
| 6.20.3.3 getStatus() | 53 |
| 6.20.3.4 setData() | 53 |
| 6.20.3.5 start() | 54 |
| 6.20.3.6 stop() | 54 |
| 6.20.4 Member Data Documentation | 54 |
| 6.20.4.1 fd | 54 |
| 6.20.4.2 i2c_address | 54 |
| 6.20.4.3 id | 54 |
| 6.20.4.4 power_state | 54 |
| 6.20.4.5 temperature | 54 |
| 7 File Documentation | 55 |
| 7.1 include/BaseSlave.h File Reference | 55 |
| 7.2 BaseSlave.h | 56 |
| 7.3 include/BusServer.h File Reference | 56 |
| 7.3.1 Macro Definition Documentation | 57 |
| 7.3.1.1 BUFFER_SIZE | 57 |
| 7.4 BusServer.h | 57 |
| 7.5 include/CO2Slave.h File Reference | 58 |
| 7.6 CO2Slave.h | 58 |
| 7.7 include/math.h File Reference | 59 |
| 7.7.1 Detailed Description | 59 |
| 7.7.2 Function Documentation | 60 |
| 7.7.2.1 add() | 60 |
| 7.7.2.2 subtract() | 60 |
| 7.8 math.h | 60 |
| 7.9 include/packets.h File Reference | 61 |
| 7.9.1 Detailed Description | 63 |
| 7.9.2 Enumeration Type Documentation | 63 |
| 7.9.2.1 PacketType | 63 |
| 7.9.2.2 SensorType | 63 |
| 7.9.3 Function Documentation | 64 |
| 7.9.3.1 __attribute__() | 64 |
| 7.9.4 Variable Documentation | 64 |

| | |
|--|----|
| 7.9.4.1 blue_state | 64 |
| 7.9.4.2 data | 64 |
| 7.9.4.3 green_state | 64 |
| 7.9.4.4 header | 65 |
| 7.9.4.5 length | 65 |
| 7.9.4.6 metadata | 65 |
| 7.9.4.7 ptype | 65 |
| 7.9.4.8 red_state | 65 |
| 7.9.4.9 sensor_id | 65 |
| 7.9.4.10 sensor_type | 66 |
| 7.9.4.11 target_state | 66 |
| 7.9.4.12 value | 66 |
| 7.10 packets.h | 66 |
| 7.11 include/RGBSlave.h File Reference | 67 |
| 7.11.1 Function Documentation | 69 |
| 7.11.1.1 __attribute__() | 69 |
| 7.11.2 Variable Documentation | 69 |
| 7.11.2.1 __attribute__ | 69 |
| 7.11.2.2 B | 69 |
| 7.11.2.3 G | 69 |
| 7.11.2.4 R | 69 |
| 7.12 RGBSlave.h | 69 |
| 7.13 include/SlaveManager.h File Reference | 70 |
| 7.14 SlaveManager.h | 70 |
| 7.15 include/TemperatureSlave.h File Reference | 71 |
| 7.15.1 Function Documentation | 72 |
| 7.15.1.1 __attribute__() | 72 |
| 7.15.2 Variable Documentation | 72 |
| 7.15.2.1 __attribute__ | 72 |
| 7.15.2.2 Temp | 72 |
| 7.16 TemperatureSlave.h | 73 |
| 7.17 README.md File Reference | 73 |
| 7.18 src/BaseSlave.cpp File Reference | 73 |
| 7.19 BaseSlave.cpp | 73 |
| 7.20 src/BusServer.cpp File Reference | 74 |
| 7.20.1 Macro Definition Documentation | 74 |
| 7.20.1.1 RGB_SLAVE_ADDRESS | 74 |
| 7.21 BusServer.cpp | 74 |
| 7.22 src/CO2Slave.cpp File Reference | 77 |
| 7.23 CO2Slave.cpp | 77 |
| 7.24 src/main.cpp File Reference | 78 |
| 7.24.1 Function Documentation | 78 |

| | |
|--|-----------|
| 7.24.1.1 main() | 78 |
| 7.25 main.cpp | 78 |
| 7.26 src/math.cpp File Reference | 79 |
| 7.26.1 Detailed Description | 79 |
| 7.26.2 Function Documentation | 80 |
| 7.26.2.1 add() | 80 |
| 7.26.2.2 subtract() | 81 |
| 7.27 math.cpp | 81 |
| 7.28 src/RGBSlave.cpp File Reference | 81 |
| 7.29 RGBSlave.cpp | 82 |
| 7.30 src/SlaveManager.cpp File Reference | 83 |
| 7.30.1 Macro Definition Documentation | 83 |
| 7.30.1.1 MASTER_ADDRESS | 83 |
| 7.31 SlaveManager.cpp | 84 |
| 7.32 src/TemperatureSlave.cpp File Reference | 85 |
| 7.33 TemperatureSlave.cpp | 85 |
| 7.34 tests/test_math.cpp File Reference | 86 |
| 7.34.1 Detailed Description | 86 |
| 7.34.2 Function Documentation | 87 |
| 7.34.2.1 TEST() [1/3] | 87 |
| 7.34.2.2 TEST() [2/3] | 87 |
| 7.34.2.3 TEST() [3/3] | 87 |
| 7.35 test_math.cpp | 87 |
| Index | 89 |

Chapter 1

Bus Bridge Server

Chapter 2

Test List

File [test_math.cpp](#)

MathTest.Add

- Verifies that the `add` function correctly computes the sum of two integers.
- Example: `add(2, 3)` should return 5.

MathTest.Subtract

- Verifies that the `subtract` function correctly computes the difference between two integers.
- Examples:
 - `subtract(10, 3)` should return 7.
 - `subtract(9, 3)` should return 6.

MathTest.SubtractNegative

- Verifies that the `subtract` function handles subtraction with negative integers correctly.
- Example: `subtract(10, -3)` should return 13.

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

| | |
|--------------------------------------|----|
| BaseSlave | 11 |
| CO2Slave | 17 |
| RGBSlave | 23 |
| TemperatureSlave | 50 |
| BusServer | 14 |
| RGBData | 22 |
| sensor_data | 28 |
| sensor_packet::sensor_data | 30 |
| sensor_header | 32 |
| sensor_heartbeat | 33 |
| sensor_metadata | 34 |
| sensor_packet | 35 |
| sensor_packet_co2 | 37 |
| sensor_packet_generic | 39 |
| sensor_packet_humidity | 40 |
| sensor_packet_light | 41 |
| sensor_packet_rgb_light | 43 |
| sensor_packet_temperature | 45 |
| SlaveManager | 46 |
| TemperatureData | 49 |

Chapter 4

Class Index

4.1 Class List

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

| | |
|---|----|
| BaseSlave | 11 |
| BusServer | 14 |
| CO2Slave | 17 |
| RGBData | 22 |
| RGBSlave | 23 |
| sensor_data | |
| 28 | |
| sensor_packet::sensor_data | |
| 30 | |
| sensor_header | |
| Header structure for sensor packets | 32 |
| sensor_heartbeat | |
| Structure for heartbeat packets | 33 |
| sensor_metadata | |
| Structure for sensor metadata, which is always included in any packet | 34 |
| sensor_packet | |
| Union structure for the entire sensor packet | 35 |
| sensor_packet_co2 | |
| Structure for CO2 sensor packets | 37 |
| sensor_packet_generic | |
| Structure for generic sensor packets | 39 |
| sensor_packet_humidity | |
| Structure for humidity sensor packets | 40 |
| sensor_packet_light | |
| Structure for light sensor packets | 41 |
| sensor_packet_rgb_light | |
| Structure for RGB light sensor packets | 43 |
| sensor_packet_temperature | |
| Structure for temperature sensor packets | 45 |
| SlaveManager | 46 |
| TemperatureData | 49 |
| TemperatureSlave | 50 |

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

| | |
|--|----|
| include/ BaseSlave.h | 55 |
| include/ BusServer.h | 56 |
| include/ CO2Slave.h | 58 |
| include/ math.h | |
| Header file for math.cpp | 59 |
| include/ packets.h | |
| Header file for packets.h | 61 |
| include/ RGBSlave.h | 67 |
| include/ SlaveManager.h | 70 |
| include/ TemperatureSlave.h | 71 |
| src/ BaseSlave.cpp | 73 |
| src/ BusServer.cpp | 74 |
| src/ CO2Slave.cpp | 77 |
| src/ main.cpp | 78 |
| src/ math.cpp | |
| Implementation of basic math operations | 79 |
| src/ RGBSlave.cpp | 81 |
| src/ SlaveManager.cpp | 83 |
| src/ TemperatureSlave.cpp | 85 |
| tests/ test_math.cpp | |
| Unit tests for mathematical operations using Google Test framework | 86 |

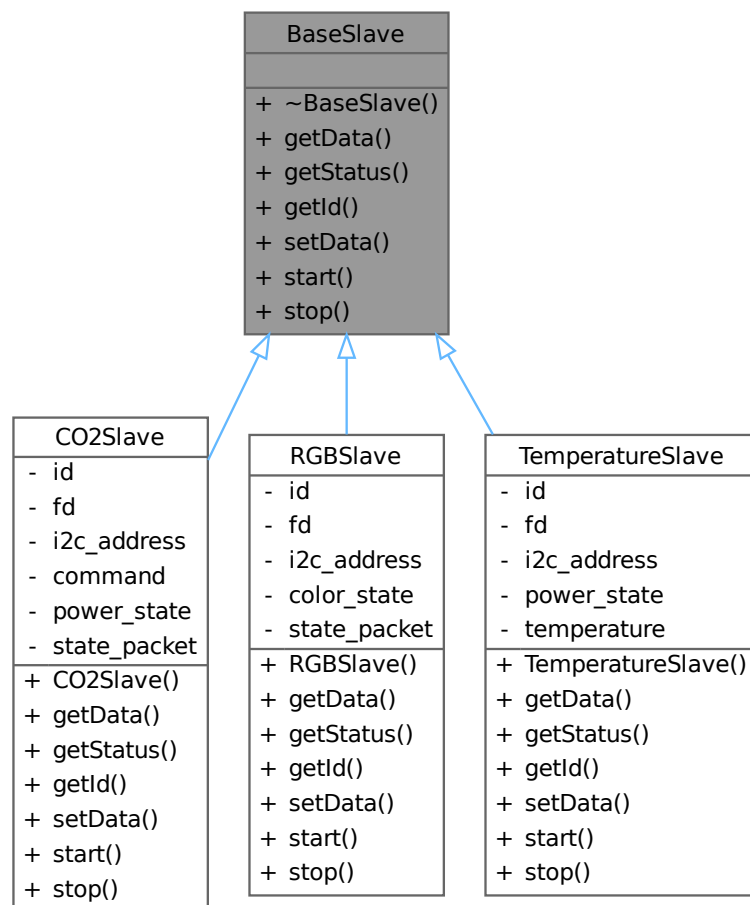
Chapter 6

Class Documentation

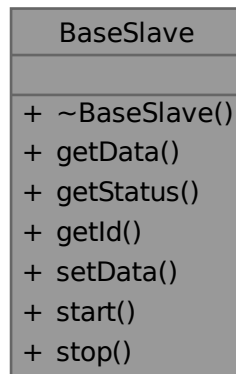
6.1 BaseSlave Class Reference

```
#include <BaseSlave.h>
```

Inheritance diagram for BaseSlave:



Collaboration diagram for BaseSlave:



Public Member Functions

- virtual [~BaseSlave](#) ()=default
- virtual const void * [getData](#) ()=0
- virtual bool [getStatus](#) ()=0
- virtual int [getId](#) ()=0
- virtual void [setData](#) (void *[data](#))=0
- virtual void [start](#) (int i2c_fd)=0
- virtual void [stop](#) ()=0

6.1.1 Detailed Description

Definition at line 6 of file [BaseSlave.h](#).

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ~BaseSlave()

```
virtual BaseSlave::~~BaseSlave ( ) [virtual], [default]
```

6.1.3 Member Function Documentation

6.1.3.1 getData()

```
virtual const void * BaseSlave::getData ( ) [pure virtual]
```

Implemented in [RGBSlave](#), [TemperatureSlave](#), and [CO2Slave](#).

6.1.3.2 getId()

```
virtual int BaseSlave::getId ( ) [pure virtual]
```

Implemented in [RGBSlave](#), [TemperatureSlave](#), and [CO2Slave](#).

6.1.3.3 getStatus()

```
virtual bool BaseSlave::getStatus ( ) [pure virtual]
```

Implemented in [RGBSlave](#), [TemperatureSlave](#), and [CO2Slave](#).

6.1.3.4 setData()

```
virtual void BaseSlave::setData (
    void * data ) [pure virtual]
```

Implemented in [RGBSlave](#), [TemperatureSlave](#), and [CO2Slave](#).

6.1.3.5 start()

```
virtual void BaseSlave::start (
    int i2c_fd ) [pure virtual]
```

Implemented in [RGBSlave](#), [TemperatureSlave](#), and [CO2Slave](#).

6.1.3.6 stop()

```
virtual void BaseSlave::stop ( ) [pure virtual]
```

Implemented in [RGBSlave](#), [TemperatureSlave](#), and [CO2Slave](#).

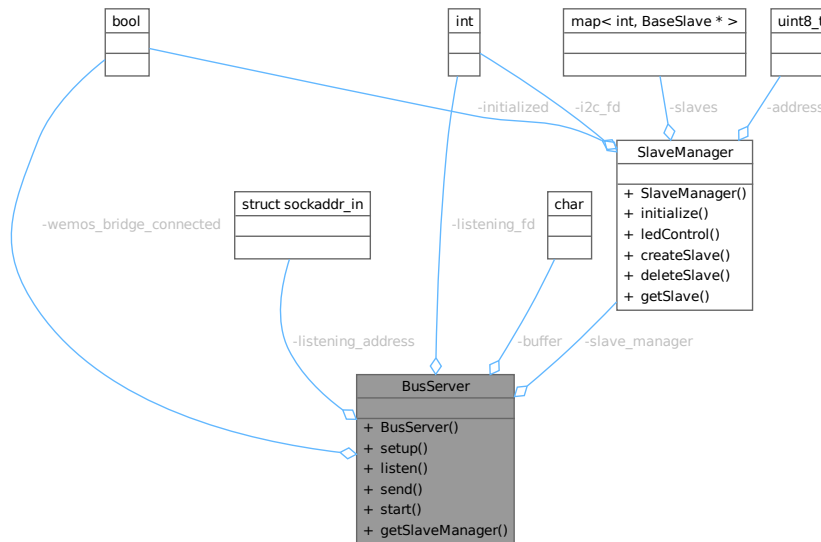
The documentation for this class was generated from the following file:

- [include/BaseSlave.h](#)

6.2 BusServer Class Reference

```
#include <BusServer.h>
```

Collaboration diagram for BusServer:



Public Member Functions

- [BusServer](#) ()
- void [setup](#) (std::string ip, int port)
Setup for the IP socket and the I2C slaves connection.
- void [listen](#) ()
Open the underlying socket for incoming connections.
- void [send](#) (struct [sensor_packet](#) *pkt, int fd)
Send a sensor packet to a connected network client.
- void [start](#) ()
Start the main loop of the [BusServer](#).
- [SlaveManager](#) & [getSlaveManager](#) ()
balls

Private Attributes

- int [listening_fd](#)
- struct sockaddr_in [listening_address](#)
- bool [wemos_bridge_connected](#) = false
- char [buffer](#) [BUFFER_SIZE]
- [SlaveManager](#) [slave_manager](#)

6.2.1 Detailed Description

Definition at line 12 of file [BusServer.h](#).

6.2.2 Constructor & Destructor Documentation

6.2.2.1 BusServer()

```
BusServer::BusServer ( ) [inline]
```

Definition at line 14 of file [BusServer.h](#).

6.2.3 Member Function Documentation

6.2.3.1 getSlaveManager()

```
SlaveManager & BusServer::getSlaveManager ( )
```

balls

Definition at line 185 of file [BusServer.cpp](#).

6.2.3.2 listen()

```
void BusServer::listen ( )
```

Open the underlying socket for incoming connections.

Exceptions

| | |
|---------------------------------|------------------------|
| <code>std::runtime_error</code> | if the listening fails |
|---------------------------------|------------------------|

Definition at line 60 of file [BusServer.cpp](#).

6.2.3.3 send()

```
void BusServer::send (
    struct sensor_packet * pkt,
    int fd )
```

Send a sensor packet to a connected network client.

Depending on the data set in the packet header, send a certain amount of data to the client.

Parameters

| | |
|------------|--|
| <i>pkt</i> | A pointer to the sensor_packet struct to send over |
| <i>fd</i> | The file descriptor to send the packet over |

Exceptions

| | |
|---------------------------------|-------------------------------------|
| <code>std::runtime_error</code> | if the sending fails for any reason |
|---------------------------------|-------------------------------------|

Definition at line 67 of file [BusServer.cpp](#).

6.2.3.4 setup()

```
void BusServer::setup (
    std::string ip,
    int port )
```

Setup for the IP socket and the I2C slaves connection.

This method will set up a socket for listening on the network and will also tell the underlying [SlaveManager](#) object to initialize its I2C bus

Parameters

| | |
|-------------|--|
| <i>ip</i> | The IP address to listen on within the network |
| <i>port</i> | The TCP port to listen on |

Exceptions

| | |
|------------------------------------|---|
| <code>std::invalid_argument</code> | if the passed IP address or port number are invalid |
| <code>std::runtime_error</code> | if the socket creation fails |

Definition at line 20 of file [BusServer.cpp](#).

6.2.3.5 start()

```
void BusServer::start ( )
```

Start the main loop of the [BusServer](#).

This will first initialize the underlying I2C connections to the directly-connected slave devices, and then start accepting and processing network clients

Definition at line 77 of file [BusServer.cpp](#).

6.2.4 Member Data Documentation

6.2.4.1 buffer

```
char BusServer::buffer[BUFFER_SIZE] [private]
```

Definition at line 60 of file [BusServer.h](#).

6.2.4.2 listening_address

```
struct sockaddr_in BusServer::listening_address [private]
```

Definition at line 58 of file [BusServer.h](#).

6.2.4.3 listening_fd

```
int BusServer::listening_fd [private]
```

Definition at line 56 of file [BusServer.h](#).

6.2.4.4 slave_manager

```
SlaveManager BusServer::slave_manager [private]
```

Definition at line 62 of file [BusServer.h](#).

6.2.4.5 wemos_bridge_connected

```
bool BusServer::wemos_bridge_connected = false [private]
```

Definition at line 59 of file [BusServer.h](#).

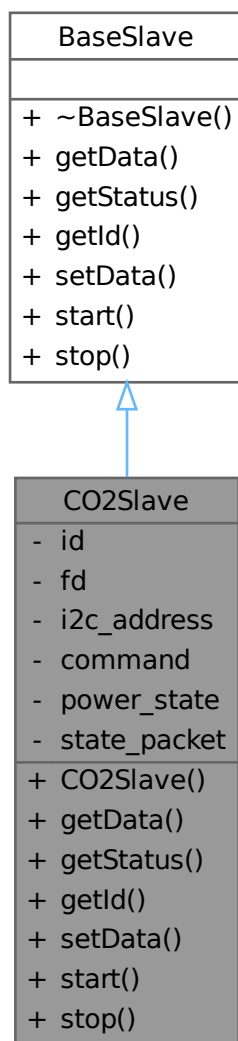
The documentation for this class was generated from the following files:

- [include/BusServer.h](#)
- [src/BusServer.cpp](#)

6.3 CO2Slave Class Reference

```
#include <CO2Slave.h>
```

Inheritance diagram for CO2Slave:



The UML class diagram illustrates the structure of the sensor system. It includes the following classes and their attributes:

- BaseSlave** (Base Class):
 - Attributes: `int`
 - Operations: `~BaseSlave()`, `get(Data)`, `get(DataOut)`, `get(S)`, `set(Data)`, `start()`, `stop()`
- CDSlave** (Derived Class):
 - Operations: `+ CDSlave()`, `get(Data)`, `get(DataOut)`, `get(S)`, `set(Data)`, `start()`, `stop()`
- PacketType**:
 - Attribute: `byte`
- SensorType**:
 - Attribute: `byte`
- unit8_t**:
 - Attributes: `sensor_type`, `length`, `sensor_id`
- sensor_header**:
 - Attributes: `byte`
- sensor_metadata**:
 - Attributes: `byte`
- float**:
 - Attributes: `blue_state`, `green_state`, `red_state`
- sensor_packet_generic**:
 - Attributes: `value`
- sensor_packet_cs2**:
 - Attributes: `metadata`
- sensor_heartbeat**:
 - Attributes: `metadata`
- sensor_packet_light**:
 - Attributes: `metadata`
- sensor_packet_temperature**:
 - Attributes: `metadata`
- sensor_packet_humidity**:
 - Attributes: `value`
- sensor_packet_rgb_light**:
 - Attributes: `value`
- sensor_packet_sensor_data**:
 - Attributes: `generic`, `cs2`, `heartbeat`, `light`, `temperature`, `humidity`, `rgb_light`
- sensor_packet**:
 - Attributes: `header`, `data`
- bool**:
 - Attribute: `power_state`

Relationships and associations are shown with blue lines and labels:

- BaseSlave** is associated with **CDSlave** via a `~BaseSlave()` association.
- BaseSlave** is associated with **sensor_packet** via a `command` association.
- BaseSlave** is associated with **unit8_t** via a `data` association.
- BaseSlave** is associated with **sensor_packet_sensor_data** via a `data` association.
- BaseSlave** is associated with **bool** via a `power_state` association.
- unit8_t** is associated with **sensor_header** via a `sensor_type` association.
- unit8_t** is associated with **sensor_metadata** via a `length` association.
- unit8_t** is associated with **sensor_metadata** via a `sensor_id` association.
- unit8_t** is associated with **float** via a `target_state` association.
- sensor_header** is associated with **sensor_packet_generic** via a `value` association.
- sensor_header** is associated with **sensor_packet_cs2** via a `metadata` association.
- sensor_header** is associated with **sensor_heartbeat** via a `metadata` association.
- sensor_header** is associated with **sensor_packet_light** via a `metadata` association.
- sensor_header** is associated with **sensor_packet_temperature** via a `metadata` association.
- sensor_header** is associated with **sensor_packet_humidity** via a `value` association.
- sensor_header** is associated with **sensor_packet_rgb_light** via a `value` association.
- sensor_metadata** is associated with **sensor_packet_generic** via a `metadata` association.
- sensor_metadata** is associated with **sensor_packet_cs2** via a `metadata` association.
- sensor_metadata** is associated with **sensor_heartbeat** via a `metadata` association.
- sensor_metadata** is associated with **sensor_packet_light** via a `metadata` association.
- sensor_metadata** is associated with **sensor_packet_temperature** via a `metadata` association.
- sensor_metadata** is associated with **sensor_packet_humidity** via a `value` association.
- sensor_metadata** is associated with **sensor_packet_rgb_light** via a `value` association.
- sensor_packet_generic** is associated with **sensor_packet_sensor_data** via a `generic` association.
- sensor_packet_cs2** is associated with **sensor_packet_sensor_data** via a `cs2` association.
- sensor_heartbeat** is associated with **sensor_packet_sensor_data** via a `heartbeat` association.
- sensor_packet_light** is associated with **sensor_packet_sensor_data** via a `light` association.
- sensor_packet_temperature** is associated with **sensor_packet_sensor_data** via a `temperature` association.
- sensor_packet_humidity** is associated with **sensor_packet_sensor_data** via a `humidity` association.
- sensor_packet_rgb_light** is associated with **sensor_packet_sensor_data** via a `rgb_light` association.
- sensor_packet_sensor_data** is associated with **sensor_packet** via a `data` association.
- sensor_packet** is associated with **unit8_t** via a `data` association.
- sensor_packet** is associated with **CDSlave** via a `data` association.
- sensor_packet** is associated with **bool** via a `power_state` association.

- `CO2Slave` (uint8_t `id`, uint8_t `i2c_address`)
- `const void * getData ()` override
- `bool getStatus ()` override
- `int getId ()` override
- `void setData (void *data)` override
- `void start (int i2c_fd)` override
- `void stop ()` override

- virtual `~BaseSlave ()`=default

- int **id**
- int **fd**
- uint8_t **i2c_address**
- uint16_t **command**
- bool **power_state** = true
- struct **sensor_packet** **state_packet**

Definition at line 7 of file CO2Slave.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 CO2Slave()

```
CO2Slave::CO2Slave (
    uint8_t id,
    uint8_t i2c_address )
```

Definition at line 5 of file [CO2Slave.cpp](#).

6.3.3 Member Function Documentation

6.3.3.1 getData()

```
const void * CO2Slave::getData ( ) [override], [virtual]
```

Implements [BaseSlave](#).

Definition at line 18 of file [CO2Slave.cpp](#).

6.3.3.2 getId()

```
int CO2Slave::getId ( ) [override], [virtual]
```

Implements [BaseSlave](#).

Definition at line 33 of file [CO2Slave.cpp](#).

6.3.3.3 getStatus()

```
bool CO2Slave::getStatus ( ) [override], [virtual]
```

Implements [BaseSlave](#).

Definition at line 31 of file [CO2Slave.cpp](#).

6.3.3.4 setData()

```
void CO2Slave::setData (
    void * data ) [override], [virtual]
```

Implements [BaseSlave](#).

Definition at line 35 of file [CO2Slave.cpp](#).

6.3.3.5 start()

```
void CO2Slave::start (
    int i2c_fd ) [override], [virtual]
```

Implements [BaseSlave](#).

Definition at line 37 of file [CO2Slave.cpp](#).

6.3.3.6 stop()

```
void CO2Slave::stop ( ) [override], [virtual]
```

Implements [BaseSlave](#).

Definition at line 39 of file [CO2Slave.cpp](#).

6.3.4 Member Data Documentation

6.3.4.1 command

```
uint16_t CO2Slave::command [private]
```

Definition at line 22 of file [CO2Slave.h](#).

6.3.4.2 fd

```
int CO2Slave::fd [private]
```

Definition at line 20 of file [CO2Slave.h](#).

6.3.4.3 i2c_address

```
uint8_t CO2Slave::i2c_address [private]
```

Definition at line 21 of file [CO2Slave.h](#).

6.3.4.4 id

```
int CO2Slave::id [private]
```

Definition at line 19 of file [CO2Slave.h](#).

6.3.4.5 power_state

```
bool CO2Slave::power_state = true [private]
```

Definition at line 24 of file [CO2Slave.h](#).

6.3.4.6 state_packet

```
struct sensor\_packet CO2Slave::state_packet [private]
```

Definition at line 25 of file [CO2Slave.h](#).

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

- include/[CO2Slave.h](#)
- src/[CO2Slave.cpp](#)

6.4 RGBData Struct Reference

```
#include <RGBSlave.h>
```

Collaboration diagram for RGBData:



Public Attributes

- `uint8_t` [R](#)
- `uint8_t` [G](#)
- `uint8_t` [B](#)

6.4.1 Detailed Description

Definition at line 7 of file [RGBSlave.h](#).

6.4.2 Member Data Documentation

6.4.2.1 B

```
uint8_t RGBData::B
```

Definition at line 8 of file [RGBSlave.h](#).

6.4.2.2 G

```
uint8_t RGBData::G
```

Definition at line 8 of file [RGBSlave.h](#).

6.4.2.3 R

```
uint8_t RGBData::R
```

Definition at line 8 of file [RGBSlave.h](#).

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

- include/[RGBSlave.h](#)

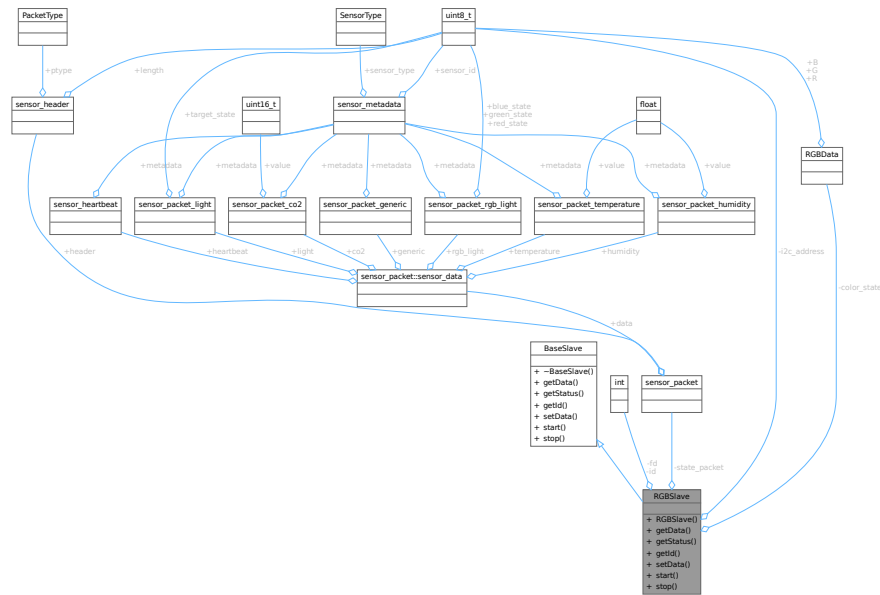
6.5 RGBSlave Class Reference

```
#include <RGBSlave.h>
```

Inheritance diagram for RGBSlave:



Collaboration diagram for RGBSlave:



Public Member Functions

- [RGBSlave](#) (uint8_t id, uint8_t i2c_address)
- const void * [getData](#) ()
- bool [getStatus](#) ()
- int [getId](#) ()
- void [setData](#) (void *data)
- void [start](#) (int i2c_fd)
- void [stop](#) ()

Public Member Functions inherited from [BaseSlave](#)

- virtual [~BaseSlave](#) ()=default

Private Attributes

- int [id](#)
- int [fd](#)
- uint8_t [i2c_address](#)
- [RGBData](#) [color_state](#)
- struct [sensor_packet](#) [state_packet](#)

6.5.1 Detailed Description

Definition at line 11 of file [RGBSlave.h](#).

6.5.2 Constructor & Destructor Documentation

6.5.2.1 RGBSlave()

```
RGBSlave::RGBSlave (
    uint8_t id,
    uint8_t i2c_address )
```

Definition at line 7 of file [RGBSlave.cpp](#).

6.5.3 Member Function Documentation

6.5.3.1 getData()

```
const void * RGBSlave::getData ( ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 15 of file [RGBSlave.cpp](#).

6.5.3.2 getId()

```
int RGBSlave::getId ( ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 42 of file [RGBSlave.cpp](#).

6.5.3.3 getStatus()

```
bool RGBSlave::getStatus ( ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 33 of file [RGBSlave.cpp](#).

6.5.3.4 setData()

```
void RGBSlave::setData (
    void * data ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 44 of file [RGBSlave.cpp](#).

6.5.3.5 start()

```
void RGBSlave::start (
    int i2c_fd ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 60 of file [RGBSlave.cpp](#).

6.5.3.6 stop()

```
void RGBSlave::stop ( ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 62 of file [RGBSlave.cpp](#).

6.5.4 Member Data Documentation

6.5.4.1 color_state

```
RGBData RGBSlave::color_state [private]
```

Definition at line 26 of file [RGBSlave.h](#).

6.5.4.2 fd

```
int RGBSlave::fd [private]
```

Definition at line 23 of file [RGBSlave.h](#).

6.5.4.3 i2c_address

```
uint8_t RGBSlave::i2c_address [private]
```

Definition at line 24 of file [RGBSlave.h](#).

6.5.4.4 id

```
int RGBSlave::id [private]
```

Definition at line 22 of file [RGBSlave.h](#).

6.5.4.5 state_packet

```
struct sensor_packet RGBSlave::state_packet [private]
```

Definition at line 28 of file [RGBSlave.h](#).

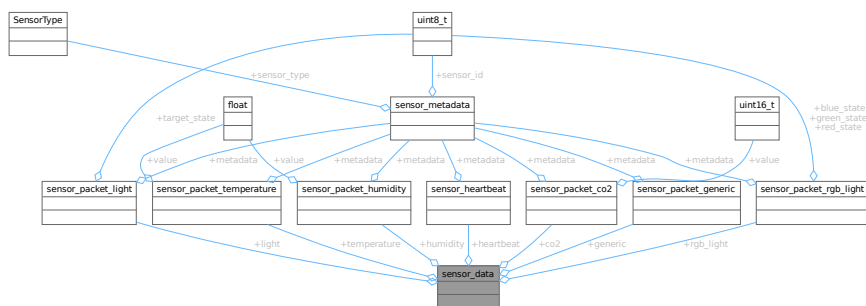
The documentation for this class was generated from the following files:

- [include/RGBSlave.h](#)
- [src/RGBSlave.cpp](#)

6.6 sensor_data Union Reference

```
#include <packets.h>
```

Collaboration diagram for `sensor_data`:



Public Attributes

- struct [sensor_heartbeat](#) heartbeat
- struct [sensor_packet_generic](#) generic
- struct [sensor_packet_temperature](#) temperature
- struct [sensor_packet_co2](#) co2
- struct [sensor_packet_humidity](#) humidity
- struct [sensor_packet_light](#) light
- struct [sensor_packet_rgb_light](#) rgb_light

6.6.1 Detailed Description

Definition at line 4 of file [packets.h](#).

6.6.2 Member Data Documentation

6.6.2.1 co2

```
struct sensor\_packet\_co2 sensor_data::co2
```

Definition at line 8 of file [packets.h](#).

6.6.2.2 generic

```
struct sensor\_packet\_generic sensor_data::generic
```

Definition at line 6 of file [packets.h](#).

6.6.2.3 heartbeat

```
struct sensor\_heartbeat sensor_data::heartbeat
```

Definition at line 5 of file [packets.h](#).

6.6.2.4 humidity

```
struct sensor\_packet\_humidity sensor_data::humidity
```

Definition at line 9 of file [packets.h](#).

6.6.2.5 light

```
struct sensor\_packet\_light sensor_data::light
```

Definition at line 10 of file [packets.h](#).

6.6.2.6 rgb_light

```
struct sensor\_packet\_rgb\_light sensor_data::rgb_light
```

Definition at line 11 of file [packets.h](#).

6.6.2.7 temperature

```
struct sensor\_packet\_temperature sensor_data::temperature
```

Definition at line 7 of file [packets.h](#).

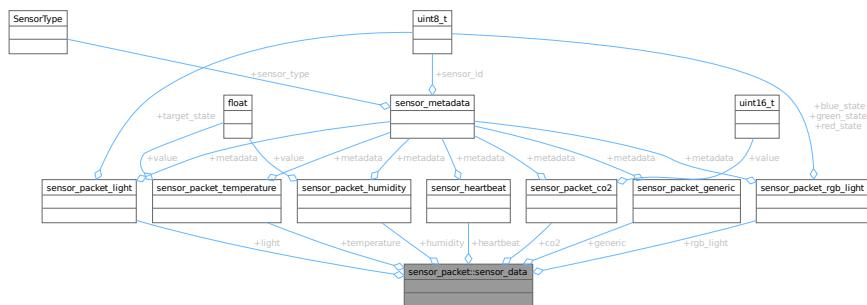
The documentation for this union was generated from the following file:

- [include/packets.h](#)

6.7 sensor_packet::sensor_data Union Reference

```
#include <packets.h>
```

Collaboration diagram for sensor_packet::sensor_data:



Public Attributes

- struct [sensor_heartbeat](#) heartbeat
- struct [sensor_packet_generic](#) generic
- struct [sensor_packet_temperature](#) temperature
- struct [sensor_packet_co2](#) co2
- struct [sensor_packet_humidity](#) humidity
- struct [sensor_packet_light](#) light
- struct [sensor_packet_rgb_light](#) rgb_light

6.7.1 Detailed Description

Definition at line 227 of file [packets.h](#).

6.7.2 Member Data Documentation

6.7.2.1 co2

```
struct sensor\_packet\_co2 sensor_packet::sensor_data::co2
```

Definition at line 231 of file [packets.h](#).

6.7.2.2 generic

```
struct sensor\_packet\_generic sensor_packet::sensor_data::generic
```

Definition at line 229 of file [packets.h](#).

6.7.2.3 heartbeat

```
struct sensor\_heartbeat sensor_packet::sensor_data::heartbeat
```

Definition at line [228](#) of file [packets.h](#).

6.7.2.4 humidity

```
struct sensor\_packet\_humidity sensor_packet::sensor_data::humidity
```

Definition at line [232](#) of file [packets.h](#).

6.7.2.5 light

```
struct sensor\_packet\_light sensor_packet::sensor_data::light
```

Definition at line [233](#) of file [packets.h](#).

6.7.2.6 rgb_light

```
struct sensor\_packet\_rgb\_light sensor_packet::sensor_data::rgb_light
```

Definition at line [234](#) of file [packets.h](#).

6.7.2.7 temperature

```
struct sensor\_packet\_temperature sensor_packet::sensor_data::temperature
```

Definition at line [230](#) of file [packets.h](#).

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

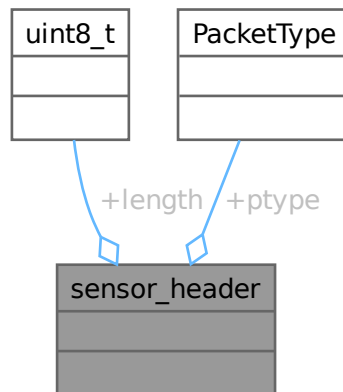
- [include/packets.h](#)

6.8 sensor_header Struct Reference

Header structure for sensor packets.

```
#include <packets.h>
```

Collaboration diagram for sensor_header:



Public Attributes

- [uint8_t length](#)
Length of the packet excluding the header.
- [PacketType ptype](#)
Type of the packet as PacketType (DATA, HEARTBEAT, etc.).

6.8.1 Detailed Description

Header structure for sensor packets.

Definition at line 40 of file [packets.h](#).

6.8.2 Member Data Documentation

6.8.2.1 length

```
uint8_t sensor_header::length
```

Length of the packet excluding the header.

Definition at line 42 of file [packets.h](#).

6.8.2.2 ptype

`PacketType sensor_header::ptype`

Type of the packet as PacketType (DATA, HEARTBEAT, etc.).

Definition at line 44 of file [packets.h](#).

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

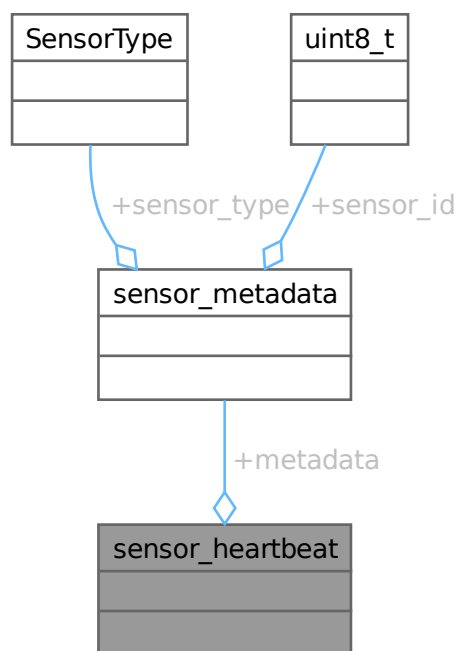
- [include/packets.h](#)

6.9 sensor_heartbeat Struct Reference

Structure for heartbeat packets.

```
#include <packets.h>
```

Collaboration diagram for sensor_heartbeat:



Public Attributes

- struct [sensor_metadata metadata](#)

6.9.1 Detailed Description

Structure for heartbeat packets.

This structure contains the type and ID of the sensor being addressed. This structure is used for heartbeat packets sent by the sensors to indicate they are still alive.

Definition at line 69 of file [packets.h](#).

6.9.2 Member Data Documentation

6.9.2.1 metadata

```
struct sensor\_metadata sensor_heartbeat::metadata
```

Definition at line 70 of file [packets.h](#).

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

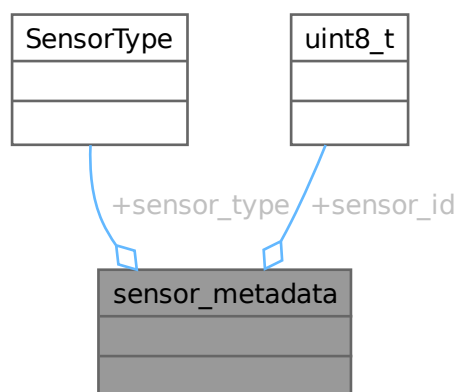
- [include/packets.h](#)

6.10 [sensor_metadata](#) Struct Reference

Structure for sensor metadata, which is always included in any packet.

```
#include <packets.h>
```

Collaboration diagram for [sensor_metadata](#):



Public Attributes

- [SensorType](#) `sensor_type`
Type of the sensor being addressed as `SensorType` (one byte)
- `uint8_t` `sensor_id`
ID of the sensor being addressed.

6.10.1 Detailed Description

Structure for sensor metadata, which is always included in any packet.

Definition at line 52 of file [packets.h](#).

6.10.2 Member Data Documentation

6.10.2.1 sensor_id

```
uint8_t sensor_metadata::sensor_id
```

ID of the sensor being addressed.

Definition at line 56 of file [packets.h](#).

6.10.2.2 sensor_type

```
SensorType sensor_metadata::sensor_type
```

Type of the sensor being addressed as `SensorType` (one byte)

Definition at line 54 of file [packets.h](#).

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

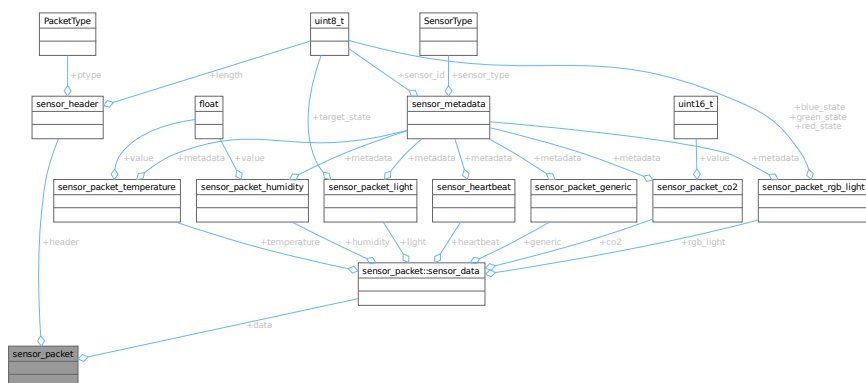
- [include/packets.h](#)

6.11 sensor_packet Struct Reference

Union structure for the entire sensor packet.

```
#include <packets.h>
```

Collaboration diagram for `sensor_packet`:



Classes

- union [sensor_data](#)

Public Attributes

- struct [sensor_header](#) header
Header of the packet containing length and type information.
- union [sensor_packet::sensor_data](#) data

6.11.1 Detailed Description

Union structure for the entire sensor packet.

This structure is used to encapsulate the different types of sensor packets that can be sent and has the shape of a valid packet.

It contains a [sensor_header](#) followed by a union of different sensor data types. The union allows for different types of sensor data to be stored in the same memory location, depending on the packet type.

Example usage:

```
sensor_packet packet;
packet.header.length = sizeof(sensor_packet_generic);
packet.header.ptype = PacketType::DATA;
packet.data.generic.metadata.sensor_type = SensorType::BUTTON;
packet.data.generic.metadata.sensor_id = 1;

// Accessing the packet data
if (packet.header.ptype == PacketType::DATA) {
    if (packet.data.generic.metadata.sensor_type == SensorType::BUTTON) {
        uint8_t sensor_id = packet.data.generic.metadata.sensor_id;
        // Process button press event for sensor_id
    }
}
```

To use this structure to request data from the dashboard, you can set the ptype to DASHBOARD_GET to indicate that you want to request data from the backend (wemos bridge). Then, you use a [sensor_packet_generic](#) to specify the type of sensor you want to request data for and the ID of that sensor.

Example: We want to request temperature data from the backend (wemos bridge) for sensor ID 1.

```
sensor_packet packet;
packet.header.length = sizeof(sensor_packet_generic);
packet.header.ptype = PacketType::DASHBOARD_GET;
packet.data.generic.metadata.sensor_type = SensorType::TEMPERATURE;
packet.data.generic.metadata.sensor_id = 1;
```

The backend (wemos bridge) will then respond with a packet of type DASHBOARD_RESPONSE containing the requested data. Following the correct type packet for this example would be a [sensor_packet_temperature](#).

Example: We want to change the color of an RGB light with ID 1 to red (255, 0, 0).

```
sensor_packet packet;
packet.header.length = sizeof(sensor_packet_rgb_light);
packet.header.ptype = PacketType::DASHBOARD_POST;
packet.data.rgb_light.metadata.sensor_type = SensorType::RGB_LIGHT;
packet.data.rgb_light.metadata.sensor_id = 1;
packet.data.rgb_light.red_state = 255;
packet.data.rgb_light.green_state = 0;
packet.data.rgb_light.blue_state = 0;
```

Note

The data field is a union that can hold different types of sensor data.

Definition at line 222 of file [packets.h](#).

6.11.2 Member Data Documentation

6.11.2.1 data

```
union sensor\_packet::sensor\_data sensor_packet::data
```

6.11.2.2 header

```
struct sensor\_header sensor_packet::header
```

Header of the packet containing length and type information.

Definition at line 224 of file [packets.h](#).

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

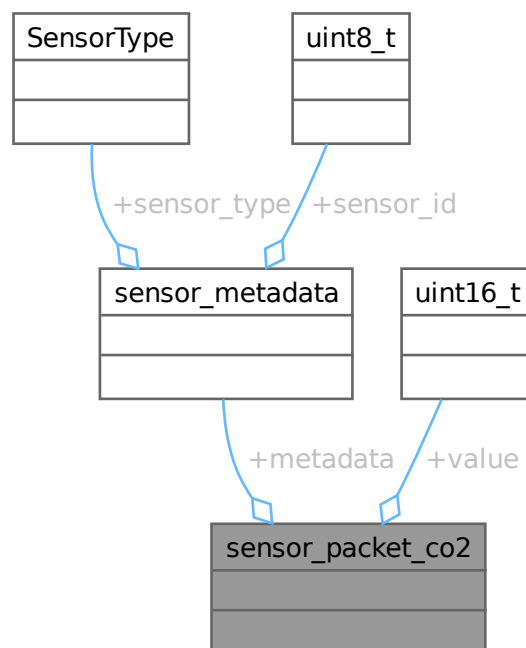
- [include/packets.h](#)

6.12 sensor_packet_co2 Struct Reference

Structure for CO2 sensor packets.

```
#include <packets.h>
```

Collaboration diagram for `sensor_packet_co2`:



Public Attributes

- struct [sensor_metadata](#) [metadata](#)
- [uint16_t](#) [value](#)

Value of the sensor reading the CO2 level represented in ppm.

6.12.1 Detailed Description

Structure for CO2 sensor packets.

This structure contains the type, ID, and value of the CO2 sensor reading.

Note

The CO2 value is represented in parts per million (ppm).

Definition at line [107](#) of file [packets.h](#).

6.12.2 Member Data Documentation

6.12.2.1 metadata

```
struct sensor\_metadata sensor_packet_co2::metadata
```

Definition at line [108](#) of file [packets.h](#).

6.12.2.2 value

```
uint16\_t sensor_packet_co2::value
```

Value of the sensor reading the CO2 level represented in ppm.

Definition at line [110](#) of file [packets.h](#).

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

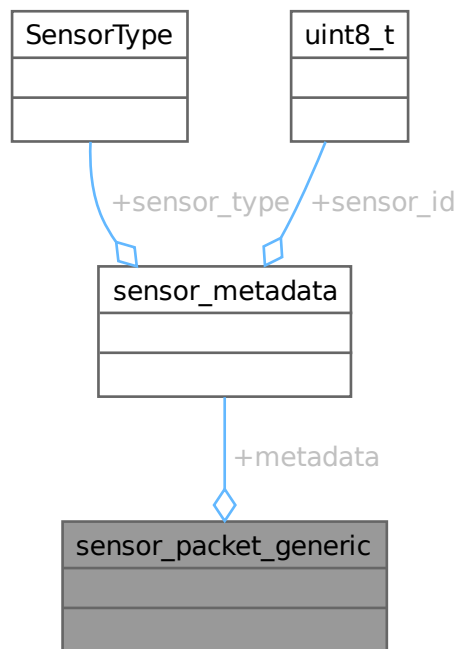
- [include/packets.h](#)

6.13 sensor_packet_generic Struct Reference

Structure for generic sensor packets.

```
#include <packets.h>
```

Collaboration diagram for sensor_packet_generic:



Public Attributes

- struct [sensor_metadata metadata](#)

6.13.1 Detailed Description

Structure for generic sensor packets.

This structure contains the type and ID of the sensor being addressed. This structure is used for generic sensor packets that do not require additional data. For example, it can be used for a simple button press event.

Definition at line 81 of file [packets.h](#).

6.13.2 Member Data Documentation

6.13.2.1 metadata

struct `sensor_metadata` `sensor_packet_generic::metadata`

Definition at line 82 of file `packets.h`.

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

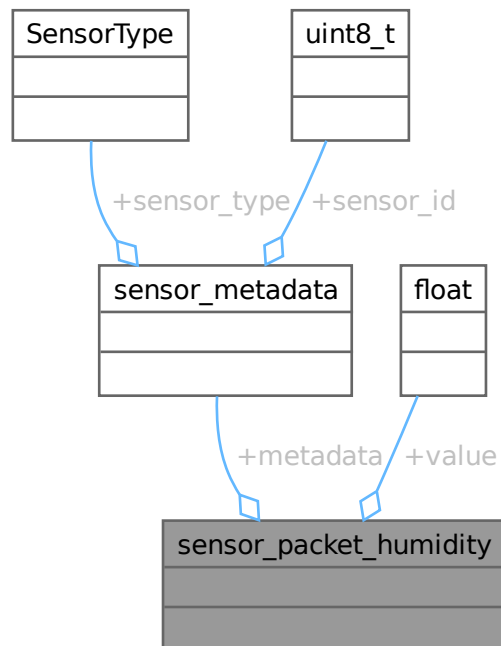
- `include/packets.h`

6.14 `sensor_packet_humidity` Struct Reference

Structure for humidity sensor packets.

```
#include <packets.h>
```

Collaboration diagram for `sensor_packet_humidity`:



Public Attributes

- struct `sensor_metadata` `metadata`
- float `value`

Value of the sensor reading the humidity level represented in percentage.

6.14.1 Detailed Description

Structure for humidity sensor packets.

This structure contains the type, ID, and value of the humidity sensor reading.

Note

The humidity value is represented in percentage.

Definition at line 120 of file [packets.h](#).

6.14.2 Member Data Documentation

6.14.2.1 metadata

```
struct sensor\_metadata sensor_packet_humidity::metadata
```

Definition at line 121 of file [packets.h](#).

6.14.2.2 value

```
float sensor_packet_humidity::value
```

Value of the sensor reading the humidity level represented in percentage.

Definition at line 123 of file [packets.h](#).

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

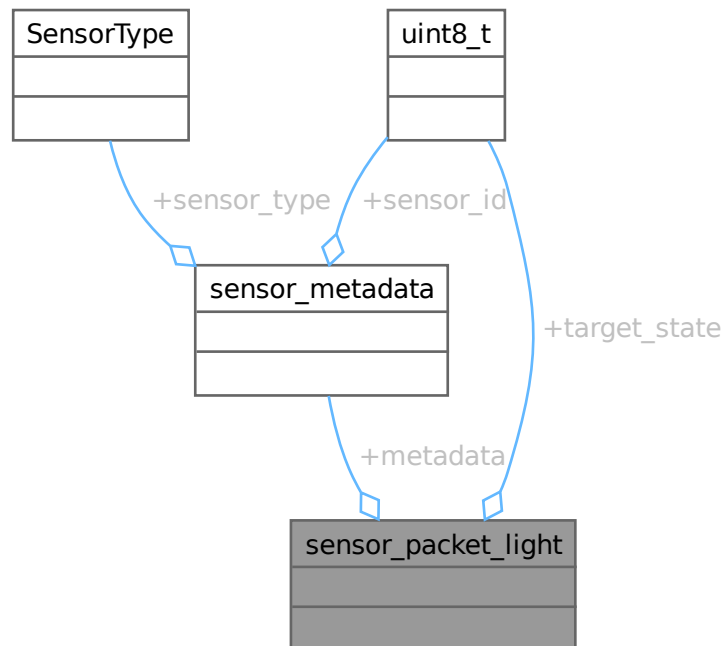
- [include/packets.h](#)

6.15 sensor_packet_light Struct Reference

Structure for light sensor packets.

```
#include <packets.h>
```

Collaboration diagram for `sensor_packet_light`:



Public Attributes

- struct `sensor_metadata metadata`
- `uint8_t target_state`

Target state of the light (on 1/off 0) represented as a boolean value.

6.15.1 Detailed Description

Structure for light sensor packets.

This structure contains the type, ID, and target state of the light/led. This structure is used for light control packets sent to the light/led.

Definition at line 133 of file `packets.h`.

6.15.2 Member Data Documentation

6.15.2.1 metadata

```
struct sensor_metadata sensor_packet_light::metadata
```

Definition at line 134 of file `packets.h`.

6.15.2.2 target_state

```
uint8_t sensor_packet_light::target_state
```

Target state of the light (on 1/off 0) represented as a boolean value.

Definition at line 136 of file [packets.h](#).

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

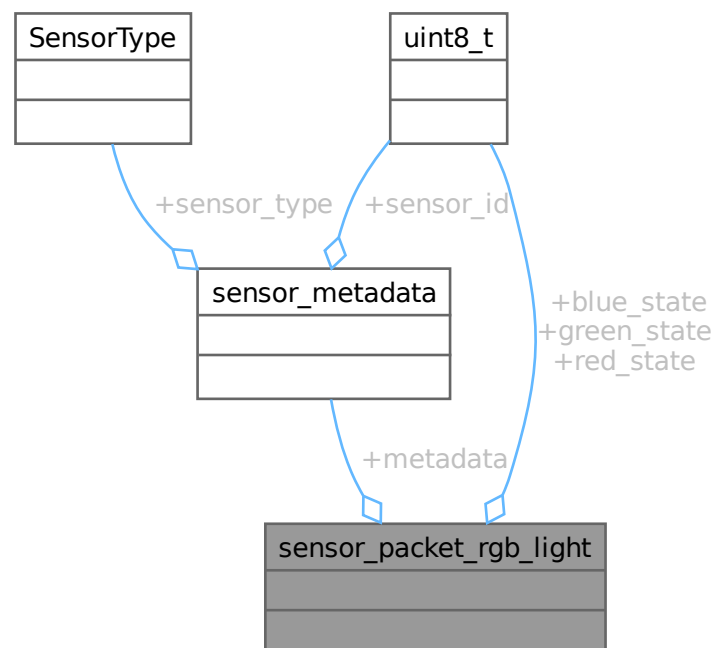
- [include/packets.h](#)

6.16 sensor_packet_rgb_light Struct Reference

Structure for RGB light sensor packets.

```
#include <packets.h>
```

Collaboration diagram for `sensor_packet_rgb_light`:



Public Attributes

- struct [sensor_metadata metadata](#)
- `uint8_t red_state`
Target state of the red color (0-255) represented as an 8-bit integer.
- `uint8_t green_state`
Target state of the green color (0-255) represented as an 8-bit integer.
- `uint8_t blue_state`
Target state of the blue color (0-255) represented as an 8-bit integer.

6.16.1 Detailed Description

Structure for RGB light sensor packets.

This structure contains the type, ID, and target color of the RGB light. This structure is used for RGB light control packets sent to the RGB light.

Note

The RGB values are represented as 8-bit integers (0-255).

Definition at line 147 of file [packets.h](#).

6.16.2 Member Data Documentation

6.16.2.1 blue_state

```
uint8_t sensor_packet_rgb_light::blue_state
```

Target state of the blue color (0-255) represented as an 8-bit integer.

Definition at line 154 of file [packets.h](#).

6.16.2.2 green_state

```
uint8_t sensor_packet_rgb_light::green_state
```

Target state of the green color (0-255) represented as an 8-bit integer.

Definition at line 152 of file [packets.h](#).

6.16.2.3 metadata

```
struct sensor_metadata sensor_packet_rgb_light::metadata
```

Definition at line 148 of file [packets.h](#).

6.16.2.4 red_state

```
uint8_t sensor_packet_rgb_light::red_state
```

Target state of the red color (0-255) represented as an 8-bit integer.

Definition at line 150 of file [packets.h](#).

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

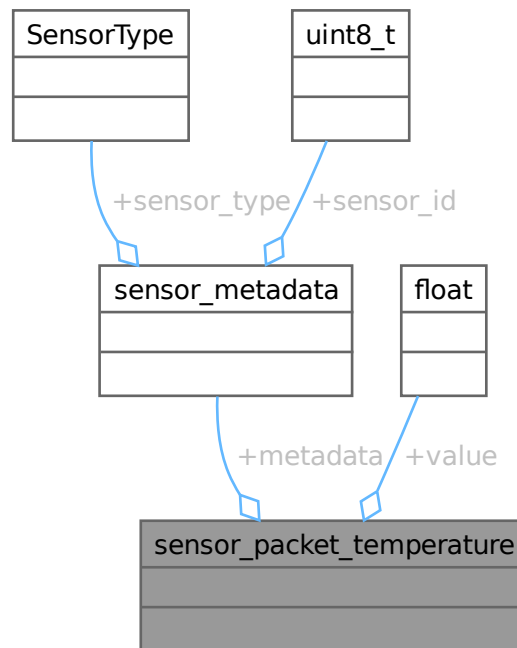
- [include/packets.h](#)

6.17 sensor_packet_temperature Struct Reference

Structure for temperature sensor packets.

```
#include <packets.h>
```

Collaboration diagram for sensor_packet_temperature:



Public Attributes

- struct [sensor_metadata metadata](#)
- float [value](#)

Value of the sensor reading the temperature represented in Celcius.

6.17.1 Detailed Description

Structure for temperature sensor packets.

This structure contains the type, ID, and value of the temperature sensor reading.

Note

The temperature value is represented in Celsius.

Definition at line 94 of file [packets.h](#).

6.17.2 Member Data Documentation

6.17.2.1 metadata

```
struct sensor\_metadata sensor_packet_temperature::metadata
```

Definition at line 95 of file [packets.h](#).

6.17.2.2 value

```
float sensor_packet_temperature::value
```

Value of the sensor reading the temperature represented in Celcius.

Definition at line 97 of file [packets.h](#).

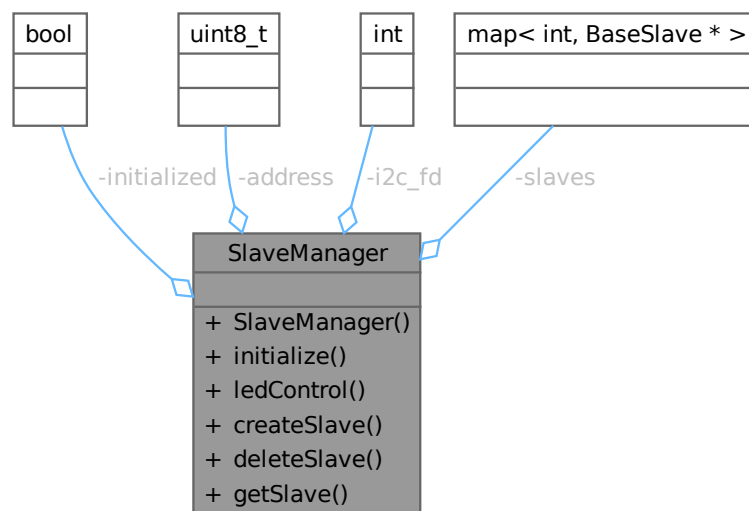
The documentation for this struct was generated from the following file:

- [include/packets.h](#)

6.18 SlaveManager Class Reference

```
#include <SlaveManager.h>
```

Collaboration diagram for SlaveManager:



Public Member Functions

- [SlaveManager](#) ()
- void [initialize](#) ()
Setup the underlying I2C bus.
- void [ledControl](#) (uint8_t led_number, uint8_t led_state)
- void [createSlave](#) ([SensorType](#) type, int i2c_address)
- void [deleteSlave](#) (uint8_t id)
Unmaps the slave from the internal mapping.
- [BaseSlave](#) * [getSlave](#) (int id)
Get a slavedevice with a given ID.

Private Attributes

- bool [initialized](#) = false
- uint8_t [address](#) = 0x01
- int [i2c_fd](#) = -1
- std::map< int, [BaseSlave](#) * > [slaves](#)

6.18.1 Detailed Description

Definition at line 9 of file [SlaveManager.h](#).

6.18.2 Constructor & Destructor Documentation

6.18.2.1 SlaveManager()

```
SlaveManager::SlaveManager ( )
```

Definition at line 15 of file [SlaveManager.cpp](#).

6.18.3 Member Function Documentation

6.18.3.1 createSlave()

```
void SlaveManager::createSlave (
    SensorType type,
    int i2c_address )
```

Definition at line 30 of file [SlaveManager.cpp](#).

6.18.3.2 deleteSlave()

```
void SlaveManager::deleteSlave (
    uint8_t id )
```

Unmaps the slave from the internal mapping.

Parameters

| | |
|-----------|-------------------------------------|
| <i>id</i> | The ID of the slave device to unmap |
|-----------|-------------------------------------|

Definition at line 47 of file [SlaveManager.cpp](#).

6.18.3.3 getSlave()

```
BaseSlave * SlaveManager::getSlave (
    int id )
```

Get a slavedevice with a given ID.

Parameters

| | |
|-----------|--|
| <i>id</i> | The ID of the slave device to retrieve |
|-----------|--|

Definition at line 56 of file [SlaveManager.cpp](#).

6.18.3.4 initialize()

```
void SlaveManager::initialize ( )
```

Setup the underlying I2C bus.

Exceptions

| | |
|---------------------------|------------------------|
| <i>std__runtime_error</i> | if the I2C setup fails |
|---------------------------|------------------------|

Definition at line 17 of file [SlaveManager.cpp](#).

6.18.3.5 ledControl()

```
void SlaveManager::ledControl (
    uint8_t led_number,
    uint8_t led_state )
```

Definition at line 28 of file [SlaveManager.cpp](#).

6.18.4 Member Data Documentation

6.18.4.1 address

```
uint8_t SlaveManager::address = 0x01 [private]
```

Definition at line 46 of file [SlaveManager.h](#).

6.18.4.2 i2c_fd

```
int SlaveManager::i2c_fd = -1 [private]
```

Definition at line 47 of file [SlaveManager.h](#).

6.18.4.3 initialized

```
bool SlaveManager::initialized = false [private]
```

Definition at line 44 of file [SlaveManager.h](#).

6.18.4.4 slaves

```
std::map<int, BaseSlave*> SlaveManager::slaves [private]
```

Definition at line 49 of file [SlaveManager.h](#).

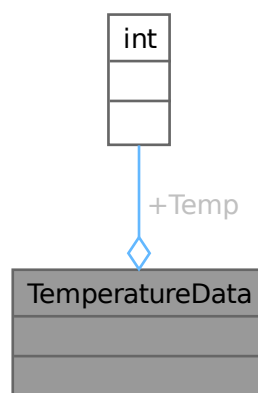
The documentation for this class was generated from the following files:

- [include/SlaveManager.h](#)
- [src/SlaveManager.cpp](#)

6.19 TemperatureData Struct Reference

```
#include <TemperatureSlave.h>
```

Collaboration diagram for TemperatureData:



Public Attributes

- int [Temp](#)

6.19.1 Detailed Description

Definition at line 6 of file [TemperatureSlave.h](#).

6.19.2 Member Data Documentation

6.19.2.1 Temp

```
int TemperatureData::Temp
```

Definition at line 7 of file [TemperatureSlave.h](#).

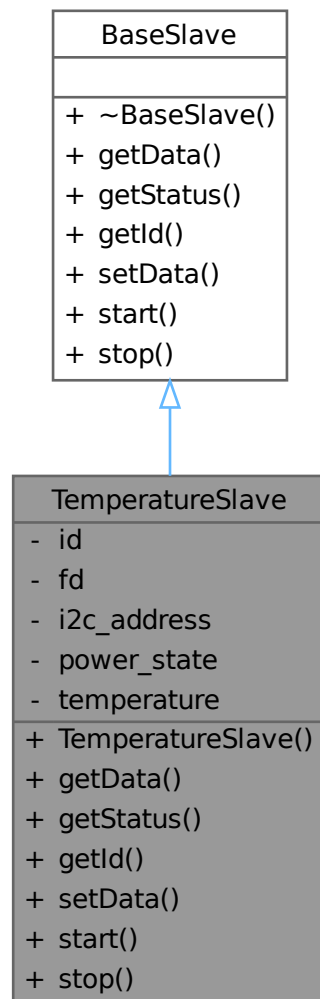
The documentation for this struct was generated from the following file:

- include/[TemperatureSlave.h](#)

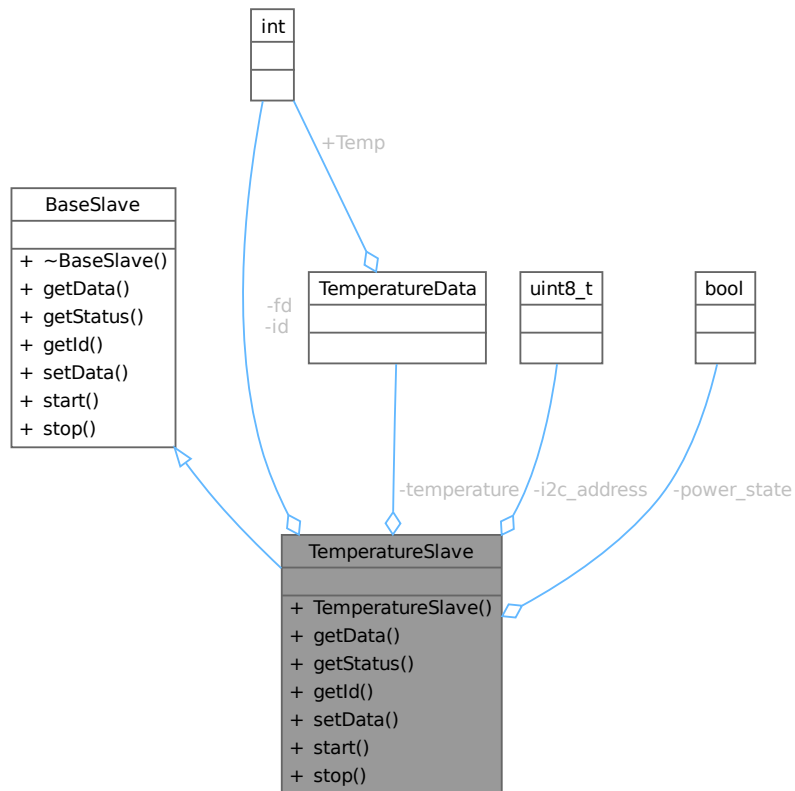
6.20 TemperatureSlave Class Reference

```
#include <TemperatureSlave.h>
```

Inheritance diagram for TemperatureSlave:



Collaboration diagram for TemperatureSlave:



Public Member Functions

- `TemperatureSlave` (`uint8_t id`, `uint8_t i2c_address`)
- `void * getData` ()
- `bool getStatus` ()
- `int getId` ()
- `void setData` (`void *data`)
- `void start` (`int i2c_fd`)
- `void stop` ()

Public Member Functions inherited from `BaseSlave`

- virtual `~BaseSlave` ()=default

Private Attributes

- `int id`
- `int fd`
- `uint8_t i2c_address`
- `bool power_state`
- `TemperatureData temperature`

6.20.1 Detailed Description

Definition at line 10 of file [TemperatureSlave.h](#).

6.20.2 Constructor & Destructor Documentation

6.20.2.1 TemperatureSlave()

```
TemperatureSlave::TemperatureSlave (
    uint8_t id,
    uint8_t i2c_address )
```

Definition at line 3 of file [TemperatureSlave.cpp](#).

6.20.3 Member Function Documentation

6.20.3.1 getData()

```
void * TemperatureSlave::getData ( ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 6 of file [TemperatureSlave.cpp](#).

6.20.3.2 getId()

```
int TemperatureSlave::getId ( ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 23 of file [TemperatureSlave.cpp](#).

6.20.3.3 getStatus()

```
bool TemperatureSlave::getStatus ( ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 14 of file [TemperatureSlave.cpp](#).

6.20.3.4 setData()

```
void TemperatureSlave::setData (
    void * data ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 25 of file [TemperatureSlave.cpp](#).

6.20.3.5 start()

```
void TemperatureSlave::start (
    int i2c_fd ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 32 of file [TemperatureSlave.cpp](#).

6.20.3.6 stop()

```
void TemperatureSlave::stop ( ) [virtual]
```

Implements [BaseSlave](#).

Definition at line 34 of file [TemperatureSlave.cpp](#).

6.20.4 Member Data Documentation

6.20.4.1 fd

```
int TemperatureSlave::fd [private]
```

Definition at line 22 of file [TemperatureSlave.h](#).

6.20.4.2 i2c_address

```
uint8_t TemperatureSlave::i2c_address [private]
```

Definition at line 23 of file [TemperatureSlave.h](#).

6.20.4.3 id

```
int TemperatureSlave::id [private]
```

Definition at line 21 of file [TemperatureSlave.h](#).

6.20.4.4 power_state

```
bool TemperatureSlave::power_state [private]
```

Definition at line 25 of file [TemperatureSlave.h](#).

6.20.4.5 temperature

```
TemperatureData TemperatureSlave::temperature [private]
```

Definition at line 26 of file [TemperatureSlave.h](#).

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

- include/[TemperatureSlave.h](#)
- src/[TemperatureSlave.cpp](#)

File Documentation

```
#include <wiringPi.h>
#include <wiringPiI2C.h>
Include dependency graph for BaseSlave.h:
```



- Generated by Doxygen

7.2 BaseSlave.h

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

```

00001 #pragma once
00002
00003 #include <wiringPi.h>
00004 #include <wiringPiI2C.h>
00005
00006 class BaseSlave {
00007 public:
00008     virtual ~BaseSlave() = default;
00009     virtual const void* getData() = 0;
00010     virtual bool getStatus() = 0;
00011     virtual int getId() = 0;
00012     virtual void setData(void* data) = 0;
00013     virtual void start(int i2c_fd) = 0;
00014     virtual void stop() = 0;
00015 };

```

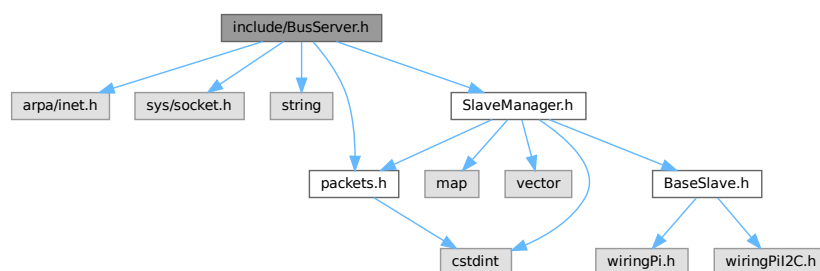
7.3 include/BusServer.h File Reference

```

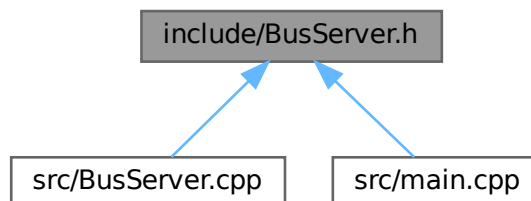
#include <arpa/inet.h>
#include <sys/socket.h>
#include <string>
#include "SlaveManager.h"
#include "packets.h"

```

Include dependency graph for BusServer.h:



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



Classes

- class [BusServer](#)

Macros

- `#define BUFFER_SIZE 1024`

7.3.1 Macro Definition Documentation

7.3.1.1 BUFFER_SIZE

```
#define BUFFER_SIZE 1024
```

Definition at line 10 of file [BusServer.h](#).

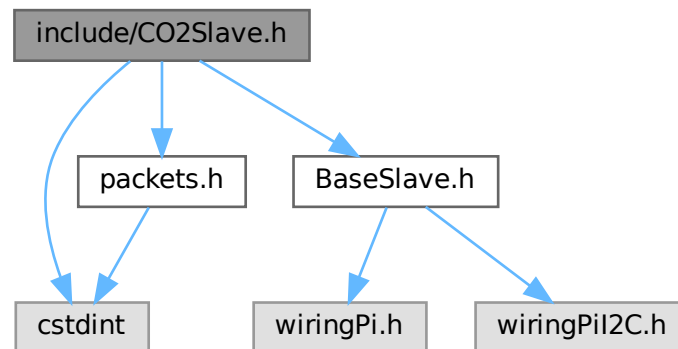
7.4 BusServer.h

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

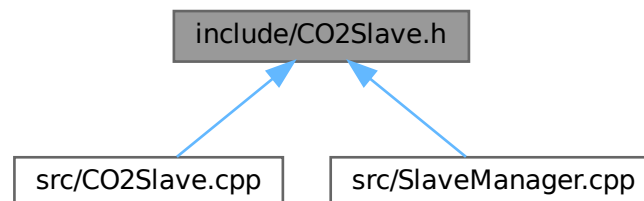
```
00001 #pragma once
00002 #include <arpa/inet.h>
00003 #include <sys/socket.h>
00004
00005 #include <string>
00006
00007 #include "SlaveManager.h"
00008 #include "packets.h"
00009
00010 #define BUFFER_SIZE 1024
00011
00012 class BusServer {
00013     public:
00014         BusServer() : listening_fd(-1) {}
00015
00025         void setup(std::string ip, int port);
00026
00031         void listen();
00032
00041         void send(struct sensor_packet* pkt, int fd);
00042
00048         void start();
00049
00053         SlaveManager& getSlaveManager();
00054
00055     private:
00056         int listening_fd;
00057
00058         struct sockaddr_in listening_address;
00059         bool wemos_bridge_connected = false;
00060         char buffer[BUFFER_SIZE];
00061
00062         SlaveManager slave_manager;
00063 };
```

7.5 include/CO2Slave.h File Reference

```
#include <cstdint>
#include "BaseSlave.h"
#include "packets.h"
Include dependency graph for CO2Slave.h:
```



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



Classes

- class [CO2Slave](#)

7.6 CO2Slave.h

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

```
00001 #pragma once
00002 #include <cstdint>
00003
00004 #include "BaseSlave.h"
```

```

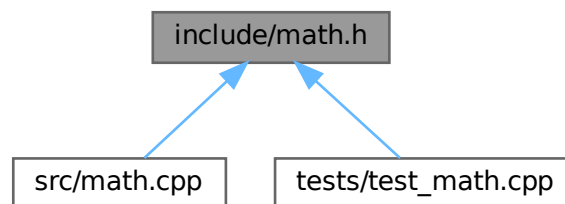
00005 #include "packets.h"
00006
00007 class CO2Slave : public BaseSlave {
00008     public:
00009         CO2Slave(uint8_t id, uint8_t i2c_address);
00010
00011         const void* getData() override;
00012         bool getStatus() override;
00013         int getId() override;
00014         void setData(void* data) override;
00015         void start(int i2c_fd) override;
00016         void stop() override;
00017
00018     private:
00019         int id;
00020         int fd;
00021         uint8_t i2c_address;
00022         uint16_t command;
00023
00024         bool power_state = true;
00025         struct sensor_packet state_packet;
00026 };

```

7.7 include/math.h File Reference

Header file for [math.cpp](#).

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



Functions

- `int add(int a, int b)`
Adds two integers.
- `int subtract(int a, int b)`
Subtracts two integers.

7.7.1 Detailed Description

Header file for [math.cpp](#).

This file contains declarations for basic math operations.

Author

Daan Breur

Definition in file [math.h](#).

7.7.2 Function Documentation

7.7.2.1 add()

```
int add (  
    int a,  
    int b )
```

Adds two integers.

Parameters

| | |
|----------|-----------------|
| <i>a</i> | First integer. |
| <i>b</i> | Second integer. |

Returns

The sum of a and b.

This function takes two integers as input and returns their sum.

Definition at line 16 of file [math.cpp](#).

7.7.2.2 subtract()

```
int subtract (  
    int a,  
    int b )
```

Subtracts two integers.

Parameters

| | |
|----------|-----------------|
| <i>a</i> | First integer. |
| <i>b</i> | Second integer. |

Returns

The difference of a and b.

This function takes two integers as input and returns the result of subtracting b from a.

Definition at line 26 of file [math.cpp](#).

7.8 math.h

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

00001

```

00008 #ifndef MATH_H
00009 #define MATH_H
00010
00011 int add(int a, int b);
00012 int subtract(int a, int b);
00013
00014 #endif

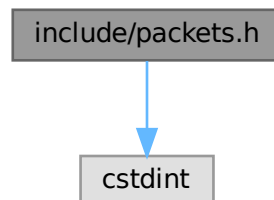
```

7.9 include/packets.h File Reference

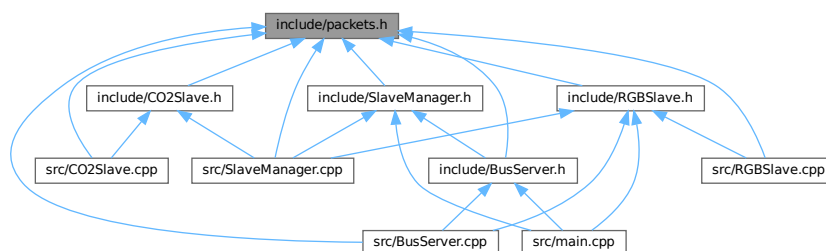
Header file for [packets.h](#).

```
#include <stdint>
```

Include dependency graph for packets.h:



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



Classes

- struct [sensor_header](#)
Header structure for sensor packets.
- struct [sensor_metadata](#)
Structure for sensor metadata, which is always included in any packet.
- struct [sensor_heartbeat](#)
Structure for heartbeat packets.
- struct [sensor_packet_generic](#)
Structure for generic sensor packets.

- struct [sensor_packet_temperature](#)
Structure for temperature sensor packets.
- struct [sensor_packet_co2](#)
Structure for CO2 sensor packets.
- struct [sensor_packet_humidity](#)
Structure for humidity sensor packets.
- struct [sensor_packet_light](#)
Structure for light sensor packets.
- struct [sensor_packet_rgb_light](#)
Structure for RGB light sensor packets.
- struct [sensor_packet](#)
Union structure for the entire sensor packet.
- union [sensor_packet::sensor_data](#)

- union [sensor_data](#)

Enumerations

- enum class [SensorType](#) : uint8_t {
 [NOOP](#) = 0 , [BUTTON](#) = 1 , [TEMPERATURE](#) = 2 , [CO2](#) = 3 ,
 [HUMIDITY](#) = 4 , [PRESSURE](#) = 5 , [LIGHT](#) = 6 , [MOTION](#) = 7 ,
 [RGB_LIGHT](#) = 8 }
- enum class [PacketType](#) : uint8_t {
 [DATA](#) = 0 , [HEARTBEAT](#) = 1 , [DASHBOARD_POST](#) = 2 , [DASHBOARD_GET](#) = 3 ,
 [DASHBOARD_RESPONSE](#) = 4 }

Functions

- struct [sensor_header __attribute__\(\(packed\)\)](#)

Variables

- uint8_t [length](#)
Length of the packet excluding the header.
- [PacketType](#) ptype
Type of the packet as PacketType (DATA, HEARTBEAT, etc.).
- [SensorType](#) sensor_type
Type of the sensor being addressed as SensorType (one byte)
- uint8_t [sensor_id](#)
ID of the sensor being addressed.
- struct [sensor_metadata](#) metadata
- float [value](#)
Value of the sensor reading the temperature represented in Celcius.
- uint8_t [target_state](#)
Target state of the light (on 1/off 0) represented as a boolean value.
- uint8_t [red_state](#)
Target state of the red color (0-255) represented as an 8-bit integer.

- `uint8_t green_state`
Target state of the green color (0-255) represented as an 8-bit integer.
- `uint8_t blue_state`
Target state of the blue color (0-255) represented as an 8-bit integer.
- `struct sensor_header header`
Header of the packet containing length and type information.
- `union sensor_data data`

7.9.1 Detailed Description

Header file for `packets.h`.

This files origin is from the Wemos project

Warning

THIS FILE MUST BE KEPT IN SYNC IN OTHER PROJECTS

Author

Daan Breur
Erynn Scholtes

Definition in file `packets.h`.

7.9.2 Enumeration Type Documentation

7.9.2.1 PacketType

```
enum class PacketType : uint8_t [strong]
```

Enumerator

| | |
|--------------------|--|
| DATA | |
| HEARTBEAT | |
| DASHBOARD_POST | |
| DASHBOARD_GET | |
| DASHBOARD_RESPONSE | |

Definition at line 27 of file `packets.h`.

7.9.2.2 SensorType

```
enum class SensorType : uint8_t [strong]
```

Enumerator

| | |
|-------------|--|
| NOOP | |
| BUTTON | |
| TEMPERATURE | |
| CO2 | |
| HUMIDITY | |
| PRESSURE | |
| LIGHT | |
| MOTION | |
| RGB_LIGHT | |

Definition at line 15 of file [packets.h](#).

7.9.3 Function Documentation

7.9.3.1 __attribute__()

```
struct sensor\_header __attribute__ (  
    (packed) )
```

7.9.4 Variable Documentation

7.9.4.1 blue_state

```
uint8_t blue_state
```

Target state of the blue color (0-255) represented as an 8-bit integer.

Definition at line 6 of file [packets.h](#).

7.9.4.2 data

```
union sensor\_data data
```

7.9.4.3 green_state

```
uint8_t green_state
```

Target state of the green color (0-255) represented as an 8-bit integer.

Definition at line 4 of file [packets.h](#).

7.9.4.4 header

```
struct sensor\_header header
```

Header of the packet containing length and type information.

Definition at line 1 of file [packets.h](#).

7.9.4.5 length

```
uint8_t length
```

Length of the packet excluding the header.

Definition at line 1 of file [packets.h](#).

7.9.4.6 metadata

```
struct sensor\_metadata metadata
```

Definition at line 0 of file [packets.h](#).

7.9.4.7 ptype

```
PacketType ptype
```

Type of the packet as PacketType (DATA, HEARTBEAT, etc.).

Definition at line 3 of file [packets.h](#).

7.9.4.8 red_state

```
uint8_t red_state
```

Target state of the red color (0-255) represented as an 8-bit integer.

Definition at line 2 of file [packets.h](#).

7.9.4.9 sensor_id

```
uint8_t sensor_id
```

ID of the sensor being addressed.

Definition at line 3 of file [packets.h](#).

7.9.4.10 sensor_type

`SensorType sensor_type`

Type of the sensor being addressed as `SensorType` (one byte)

Definition at line 1 of file [packets.h](#).

7.9.4.11 target_state

`uint8_t target_state`

Target state of the light (on 1/off 0) represented as a boolean value.

Definition at line 2 of file [packets.h](#).

7.9.4.12 value

`float value`

Value of the sensor reading the temperature represented in Celcius.

Value of the sensor reading the humidity level represented in percentage.

Value of the sensor reading the CO2 level represented in ppm.

Definition at line 2 of file [packets.h](#).

7.10 packets.h

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

```
00001
00010 #ifndef PACKETS_H
00011 #define PACKETS_H
00012
00013 #include <stdint>
00014
00015 enum class SensorType : uint8_t {
00016     NOOP = 0,
00017     BUTTON = 1,
00018     TEMPERATURE = 2,
00019     CO2 = 3,
00020     HUMIDITY = 4,
00021     PRESSURE = 5,
00022     LIGHT = 6,
00023     MOTION = 7,
00024     RGB_LIGHT = 8,
00025 };
00026
00027 enum class PacketType : uint8_t {
00028     DATA = 0,
00029     HEARTBEAT = 1,
00030     DASHBOARD_POST = 2,
00031     DASHBOARD_GET = 3,
00032     DASHBOARD_RESPONSE = 4
00033 };
00034
00040 struct sensor_header {
00042     uint8_t length;
00044     PacketType ptype;
00045 } __attribute__((packed));
00046
00052 struct sensor_metadata {
```

```

00054     SensorType sensor_type;
00056     uint8_t sensor_id;
00057 } __attribute__((packed));
00058
00059 // Specific packet structures (ensure alignment/packing matches expected format)
00060
00069 struct sensor_heartbeat {
00070     struct sensor_metadata metadata;
00071 } __attribute__((packed));
00072
00081 struct sensor_packet_generic {
00082     struct sensor_metadata metadata;
00083     // /** @brief Whether the sensor did or did not trigger */
00084     // bool value;
00085 } __attribute__((packed));
00086
00094 struct sensor_packet_temperature {
00095     struct sensor_metadata metadata;
00097     float value;
00098 } __attribute__((packed));
00099
00107 struct sensor_packet_co2 {
00108     struct sensor_metadata metadata;
00110     uint16_t value;
00111 } __attribute__((packed));
00112
00120 struct sensor_packet_humidity {
00121     struct sensor_metadata metadata;
00123     float value;
00124 } __attribute__((packed));
00125
00133 struct sensor_packet_light {
00134     struct sensor_metadata metadata;
00136     uint8_t target_state;
00137 } __attribute__((packed));
00138
00147 struct sensor_packet_rgb_light {
00148     struct sensor_metadata metadata;
00150     uint8_t red_state;
00152     uint8_t green_state;
00154     uint8_t blue_state;
00155 } __attribute__((packed));
00156 // --- End Structures ---
00157
00222 struct sensor_packet {
00224     struct sensor_header header;
00225
00227     union sensor_data {
00228         struct sensor_heartbeat heartbeat;
00229         struct sensor_packet_generic generic;
00230         struct sensor_packet_temperature temperature;
00231         struct sensor_packet_co2 co2;
00232         struct sensor_packet_humidity humidity;
00233         struct sensor_packet_light light;
00234         struct sensor_packet_rgb_light rgb_light;
00235     } data;
00236 } __attribute__((packed));
00237
00238 #endif // PACKETS_H

```

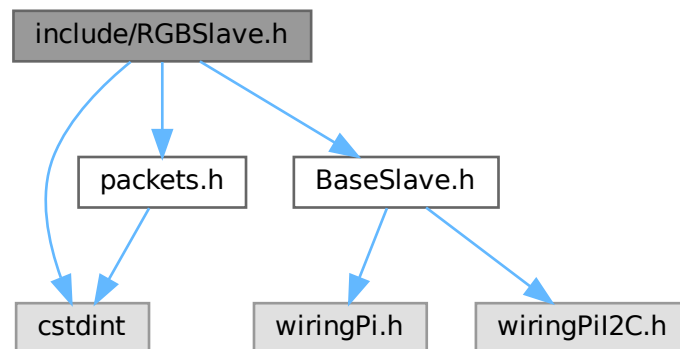
7.11 include/RGBSlave.h File Reference

```

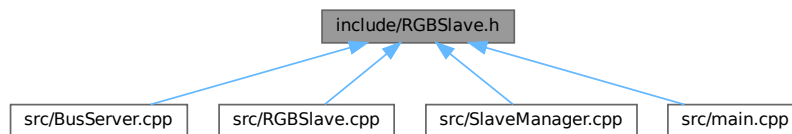
#include <stdint>
#include "BaseSlave.h"
#include "packets.h"

```

Include dependency graph for RGBSlave.h:



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



Classes

- struct [RGBData](#)
- class [RGBSlave](#)

Functions

- struct [RGBData](#) `__attribute__((packed))`

Variables

- `uint8_t` [R](#)
- `uint8_t` [G](#)
- `uint8_t` [B](#)
- [RGBSlave](#) `__attribute__((packed))`

7.11.1 Function Documentation

7.11.1.1 __attribute__()

```
struct RGBData __attribute__ (
    (packed) )
```

7.11.2 Variable Documentation

7.11.2.1 __attribute__

```
struct sensor_packet __attribute__
```

7.11.2.2 B

```
uint8_t B
```

Definition at line 0 of file [RGBSlave.h](#).

7.11.2.3 G

```
uint8_t G
```

Definition at line 0 of file [RGBSlave.h](#).

7.11.2.4 R

```
uint8_t R
```

Definition at line 0 of file [RGBSlave.h](#).

7.12 RGBSlave.h

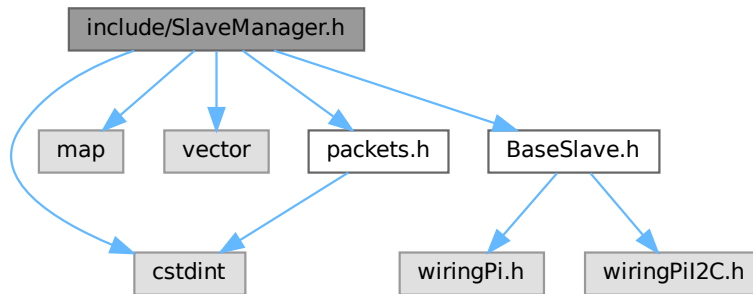
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <stdint>
00003
00004 #include "BaseSlave.h"
00005 #include "packets.h"
00006
00007 struct RGBData {
00008     uint8_t R, G, B;
00009 } __attribute__((packed));
00010
00011 class RGBSlave : public BaseSlave {
00012     public:
00013         RGBSlave(uint8_t id, uint8_t i2c_address);
00014         const void* getData();
00015         bool getStatus();
00016         int getId();
00017         void setData(void* data);
00018         void start(int i2c_fd);
00019         void stop();
00020
00021     private:
00022         int id;
00023         int fd;
00024         uint8_t i2c_address;
00025
00026         RGBData color_state;
00027
00028         struct sensor_packet state_packet;
00029     };
```

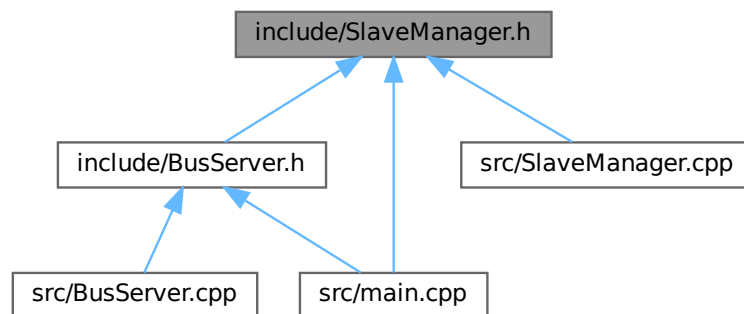
7.13 include/SlaveManager.h File Reference

```
#include <cstdint>
#include <map>
#include <vector>
#include "BaseSlave.h"
#include "packets.h"
```

Include dependency graph for SlaveManager.h:



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



Classes

- class [SlaveManager](#)

7.14 SlaveManager.h

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

```
00001 #pragma once
```



```

00002 #include <cstdint>
00003 #include <map>
00004 #include <vector>
00005
00006 #include "BaseSlave.h"
00007 #include "packets.h"
00008
00009 class SlaveManager {
00010 public:
00011     SlaveManager();
00012
00017     void initialize();
00018
00019     void ledControl(uint8_t led_number, uint8_t led_state);
00020
00021     /*
00022      * @brief Creates a new slave device into the internal mapping
00023      * @details Assigns a pre-known SensorType and ID to an I2C device, and then opens the I2C
00024      * connection to it
00025      * @param type The SensorType of the sensor, as defined in packets.h
00026      * @param id The ID of the slave device
00027      * @param i2c_address The I2C address of the slave device
00028      */
00029     void createSlave(SensorType type, int i2c_address);
00030
00035     void deleteSlave(uint8_t id);
00036
00041     BaseSlave* getSlave(int id);
00042
00043 private:
00044     bool initialized = false;
00045
00046     uint8_t address = 0x01;
00047     int i2c_fd = -1;
00048
00049     std::map<int, BaseSlave*> slaves; // maps ID to BaseSlave ptr
00050 };

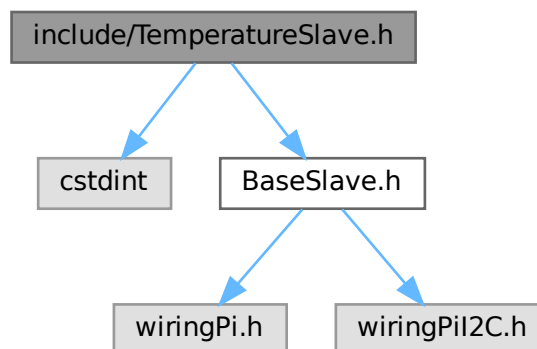
```

7.15 include/TemperatureSlave.h File Reference

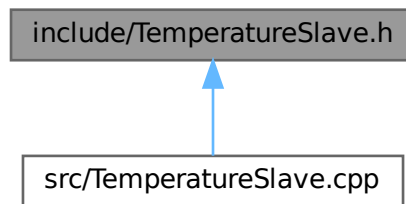
```
#include <cstdint>
```

```
#include "BaseSlave.h"
```

Include dependency graph for TemperatureSlave.h:



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



Classes

- struct [TemperatureData](#)
- class [TemperatureSlave](#)

Functions

- struct [TemperatureData](#) [__attribute__](#) ((packed))

Variables

- int [Temp](#)
- [TemperatureSlave](#) [__attribute__](#)

7.15.1 Function Documentation

7.15.1.1 [__attribute__](#)()

```
struct TemperatureData \_\_attribute\_\_ (  
    (packed) )
```

7.15.2 Variable Documentation

7.15.2.1 [__attribute__](#)

[TemperatureSlave](#) [__attribute__](#)

7.15.2.2 [Temp](#)

```
int Temp
```

Definition at line 0 of file [TemperatureSlave.h](#).

7.16 TemperatureSlave.h

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

```

00001 #pragma once
00002 #include <stdint>
00003
00004 #include "BaseSlave.h"
00005
00006 struct TemperatureData {
00007     int Temp;
00008 } __attribute__((packed));
00009
00010 class TemperatureSlave : public BaseSlave {
00011     public:
00012         TemperatureSlave(uint8_t id, uint8_t i2c_address);
00013         void* getData();
00014         bool getStatus();
00015         int getId();
00016         void setData(void* data);
00017         void start(int i2c_fd);
00018         void stop();
00019
00020     private:
00021         int id;
00022         int fd;
00023         uint8_t i2c_address;
00024
00025         bool power_state;
00026         TemperatureData temperature;
00027 };

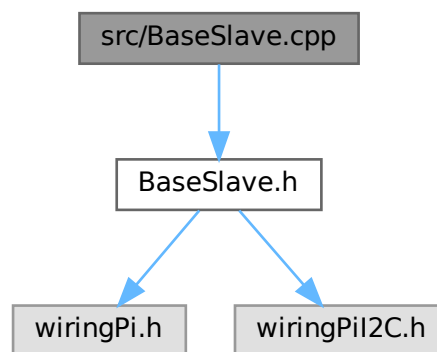
```

7.17 README.md File Reference

7.18 src/BaseSlave.cpp File Reference

```
#include "BaseSlave.h"
```

Include dependency graph for BaseSlave.cpp:



7.19 BaseSlave.cpp

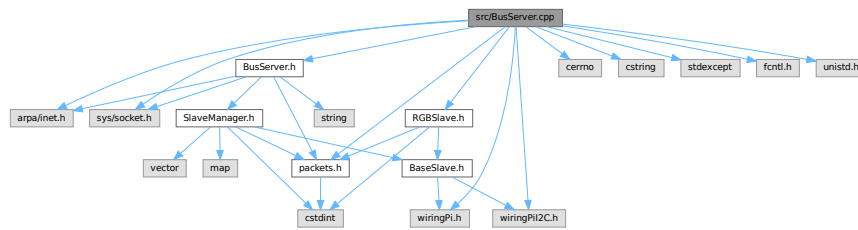
[Go to the documentation of this file.](#)

```
00001 #include "BaseSlave.h"
```

7.20 src/BusServer.cpp File Reference

```
#include "BusServer.h"
#include <arpa/inet.h>
#include <sys/socket.h>
#include <cerrno>
#include <cstring>
#include <stdexcept>
#include <fcntl.h>
#include <unistd.h>
#include "RGBSlave.h"
#include "packets.h"
#include "wiringPi.h"
#include "wiringPiI2C.h"
```

Include dependency graph for BusServer.cpp:



Macros

- `#define RGB_SLAVE_ADDRESS 0x69`

7.20.1 Macro Definition Documentation

7.20.1.1 RGB_SLAVE_ADDRESS

```
#define RGB_SLAVE_ADDRESS 0x69
```

Definition at line 18 of file [BusServer.cpp](#).

7.21 BusServer.cpp

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

```
00001 #include "BusServer.h"
00002
00003 #include <arpa/inet.h>
00004 #include <sys/socket.h>
00005
00006 #include <cerrno>
00007 #include <cstring>
00008 #include <stdexcept>
00009
00010 #include <fcntl.h>
00011 #include <unistd.h>
00012
00013 #include "RGBSlave.h"
00014 #include "packets.h"
```

```

00015 #include "wiringPi.h"
00016 #include "wiringPiI2C.h"
00017
00018 #define RGB_SLAVE_ADDRESS 0x69
00019
00020 void BusServer::setup(std::string ip, int port) {
00021     // slave_manager.initialize();
00022
00023     memset(&listening_address, 0, sizeof(listening_address));
00024
00025     if (port <= 0 || port > 65535) {
00026         throw std::invalid_argument("invalid port passed");
00027     }
00028
00029     if (0 == inet_aton(ip.c_str(), &listening_address.sin_addr)) {
00030         perror("inet_aton()");
00031         throw std::invalid_argument("invalid IP address passed");
00032     }
00033
00034     listening_address.sin_family = AF_INET;
00035     listening_address.sin_port = htons(port);
00036
00037     listening_fd = socket(AF_INET, SOCK_STREAM | SOCK_NONBLOCK, 0);
00038     if (-1 == listening_fd) {
00039         perror("socket()");
00040         throw std::runtime_error("failed to create socket");
00041     }
00042
00043     {
00044         int one = 1;
00045         if (-1 ==
00046             setsockopt(listening_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &one, sizeof(one))) {
00047             perror("setsockopt()");
00048             throw std::runtime_error("failed to set socket options");
00049         }
00050     }
00051
00052     socklen_t len = sizeof(listening_address);
00053
00054     if (-1 == bind(listening_fd, (struct sockaddr*)&listening_address, len)) {
00055         perror("bind()");
00056         throw std::runtime_error("failed to bind socket");
00057     }
00058 }
00059
00060 void BusServer::listen() {
00061     if (-1 == ::listen(listening_fd, 8)) {
00062         perror("listen()");
00063         throw std::runtime_error("failed to listen on socket");
00064     }
00065 }
00066
00067 void BusServer::send(struct sensor_packet* packet_ptr, int fd) {
00068     if (!wemos_bridge_connected) {
00069         throw std::runtime_error("no longer connected to socket");
00070     }
00071     if (-1 == ::send(fd, packet_ptr, sizeof(struct sensor_header) + packet_ptr->header.length, 0)) {
00072         perror("send()");
00073         throw std::runtime_error("failed to send on socket");
00074     }
00075 }
00076
00077 void BusServer::start() {
00078     // start off by mapping pre-known addresses to ID's
00079     listen();
00080
00081     int the_fd = -1;
00082
00083     while (true) {
00084         char buffer[32] = {0};
00085         bool net_request = false;
00086
00087         int new_fd = -1;
00088
00089         {
00090             struct sockaddr_in client_address = {0};
00091             socklen_t client_address_length = sizeof(client_address);
00092
00093             new_fd = accept4(listening_fd, (struct sockaddr*)&client_address,
00094                             &client_address_length, SOCK_NONBLOCK);
00095
00096             if (-1 == new_fd) {
00097
00098                 int err = errno;
00099                 switch (err) {
00100                     case EWOULDBLOCK:

```

```

00102             // no client has tried to connect; no big deal
00103             break;
00104
00105             default:
00106                 perror("accept4()");
00107                 throw std::runtime_error("accept4() failed");
00108                 break;
00109         }
00110     } else {
00111         the_fd = new_fd;
00112         //printf("new fd: %d\nthe fd: %d\n", new_fd, the_fd);
00113     }
00114
00115
00116     if (the_fd != -1) {
00117         int err;
00118         int recvd = recv(the_fd, buffer, sizeof(buffer), SOCK_NONBLOCK);
00119         if (0 == recvd) {
00120             close(the_fd);
00121             the_fd = -1;
00122             continue;
00123         }
00124
00125         if (-1 == recvd)
00126             err = errno;
00127         //printf("%d\n", err);
00128         //perror("recv balls");
00129         if (-1 == recvd && err != EAGAIN) {
00130             perror("recv()");
00131             usleep(100000);
00132             throw std::runtime_error("reading from client socket failed for some reason");
00133         } else if (errno != EAGAIN) {
00134             //printf("%d\n", recvd);
00135             } else {
00136                 // printf("%d\n", recvd);
00137                 if (recvd > 0)
00138                     net_request = true;
00139
00140                 //perror("recv()[1]");
00141                 usleep(100000);
00142             }
00143         }
00144     }
00145
00146     if (net_request) {
00147         struct sensor_packet* pkt_ptr = (struct sensor_packet*)buffer;
00148         BaseSlave* the_slave = slave_manager.getSlave(pkt_ptr->data.generic.metadata.sensor_id);
00149         SensorType the_sensor_type = pkt_ptr->data.generic.metadata.sensor_type;
00150         uint8_t values[8] = {0};
00151
00152         switch (pkt_ptr->header.ptype) {
00153             case PacketType::DASHBOARD_POST:
00154                 switch (the_sensor_type) {
00155                     case SensorType::LIGHT:
00156                         values[0] = pkt_ptr->data.light.target_state;
00157
00158                         the_slave->setData(values);
00159                         break;
00160
00161                     case SensorType::RGB_LIGHT:
00162                         struct RGBData rgb_data = {.R = pkt_ptr->data.rgb_light.red_state,
00163                                                     .G = pkt_ptr->data.rgb_light.green_state,
00164                                                     .B = pkt_ptr->data.rgb_light.blue_state};
00165
00166                         the_slave->setData(&rgb_data);
00167
00168                         break;
00169                 }
00170                 break;
00171
00172             case PacketType::DASHBOARD_GET:
00173                 struct sensor_packet state_ptr;
00174                 memcpy(&state_ptr, the_slave->getData(), sizeof(struct sensor_packet));
00175
00176                 state_ptr.header.ptype = PacketType::DASHBOARD_RESPONSE;
00177
00178                 send(&state_ptr, new_fd);
00179                 break;
00180             }
00181         }
00182     }
00183 }
00184
00185 SlaveManager& BusServer::getSlaveManager() {
00186     return slave_manager;
00187 }

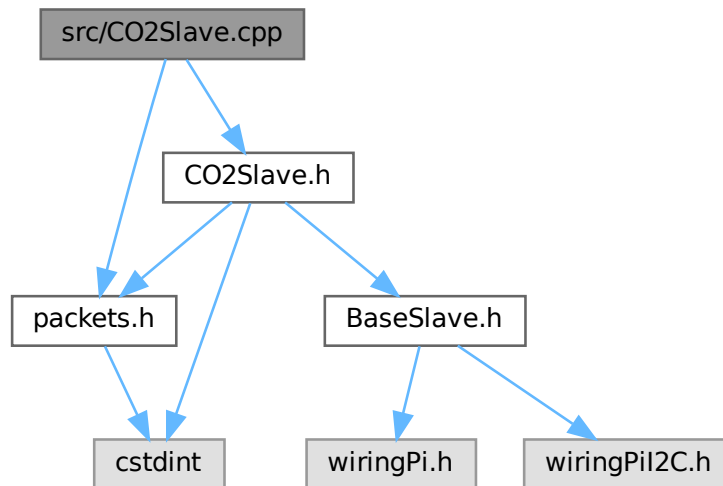
```

7.22 src/CO2Slave.cpp File Reference

```
#include "CO2Slave.h"
```

```
#include "packets.h"
```

Include dependency graph for CO2Slave.cpp:



7.23 CO2Slave.cpp

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

```

00001 #include "CO2Slave.h"
00002
00003 #include "packets.h"
00004
00005 CO2Slave::CO2Slave(uint8_t id, uint8_t i2c_address) : id(id) {
00006     state_packet.header.ptype = PacketType::DATA;
00007     state_packet.header.length = sizeof(struct sensor_packet_co2);
00008
00009     state_packet.data.rgb_light.metadata.sensor_id = id;
00010     state_packet.data.rgb_light.metadata.sensor_type = SensorType::CO2;
00011     command = 0x2003;
00012     uint8_t* command_ptr = (uint8_t*)&command;
00013
00014     wiringPiI2CWrite(fd, command_ptr[0]);
00015     wiringPiI2CWrite(fd, command_ptr[1]);
00016 }
00017
00018 const void* CO2Slave::getData() {
00019     command = 0x2008;
00020     uint8_t* command_ptr = (uint8_t*)&command;
00021     wiringPiI2CWrite(fd, command_ptr[0]);
00022     wiringPiI2CWrite(fd, command_ptr[1]);
00023     uint8_t* value_ptr = (uint8_t*)&state_packet.data.co2.value;
00024
00025     value_ptr[1] = wiringPiI2CRead(fd);
00026     value_ptr[0] = wiringPiI2CRead(fd);
00027     uint8_t empty = wiringPiI2CRead(fd);
00028     return &state_packet;
00029 }
00030
00031 bool CO2Slave::getStatus() { return power_state; }
00032
00033 int CO2Slave::getId() { return id; }

```

```

00034
00035 void CO2Slave::setData(void* data) { return; }
00036
00037 void CO2Slave::start(int i2c_fd) { fd = i2c_fd; }
00038
00039 void CO2Slave::stop() { fd = -1; }

```

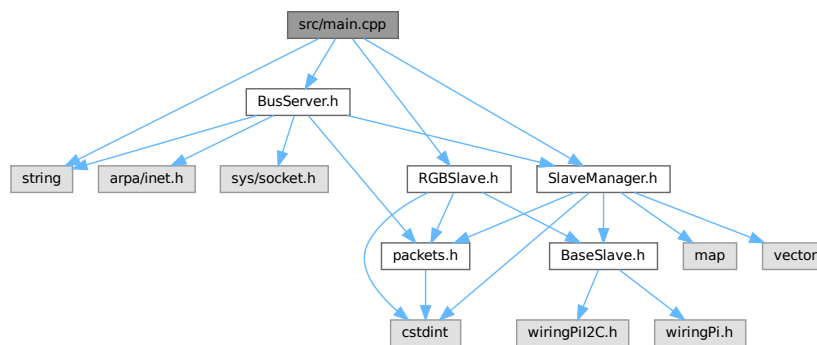
7.24 src/main.cpp File Reference

```

#include <string>
#include "BusServer.h"
#include "RGBSlave.h"
#include "SlaveManager.h"

```

Include dependency graph for main.cpp:



Functions

- int [main](#) ()

7.24.1 Function Documentation

7.24.1.1 main()

```
int main ( )
```

Definition at line 7 of file [main.cpp](#).

7.25 main.cpp

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

```

00001 #include <string>
00002
00003 #include "BusServer.h"
00004 #include "RGBSlave.h"
00005 #include "SlaveManager.h"
00006
00007 int main() {
00008     BusServer bus_server;

```



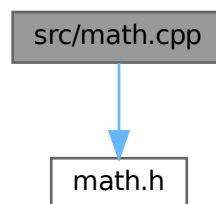
```
00009     bus_server.setup("0.0.0.0", 5000);
00010
00011     SlaveManager& slave_manager_ref = bus_server.getSlaveManager();
00012
00013     {
00014         int fd = wiringPiI2CSetup(0x69);
00015
00016         slave_manager_ref.createSlave(SensorType::RGB_LIGHT, 0x69);
00017         slave_manager_ref.getSlave(0x69)->start(fd);
00018     }
00019
00020     bus_server.start();
00021 }
```

7.26 src/math.cpp File Reference

Implementation of basic math operations.

```
#include "math.h"
```

Include dependency graph for math.cpp:



Functions

- int [add](#) (int a, int b)
Adds two integers.
- int [subtract](#) (int a, int b)
Subtracts two integers.

7.26.1 Detailed Description

Implementation of basic math operations.

Author

Daan Breur

Definition in file [math.cpp](#).

7.26.2 Function Documentation

7.26.2.1 add()

```
int add (  
    int a,  
    int b )
```

Adds two integers.

Parameters

| | |
|----------|-----------------|
| <i>a</i> | First integer. |
| <i>b</i> | Second integer. |

Returns

The sum of a and b.

This function takes two integers as input and returns their sum.

Definition at line 16 of file [math.cpp](#).

7.26.2.2 subtract()

```
int subtract (
    int a,
    int b )
```

Subtracts two integers.

Parameters

| | |
|----------|-----------------|
| <i>a</i> | First integer. |
| <i>b</i> | Second integer. |

Returns

The difference of a and b.

This function takes two integers as input and returns the result of subtracting b from a.

Definition at line 26 of file [math.cpp](#).

7.27 math.cpp

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

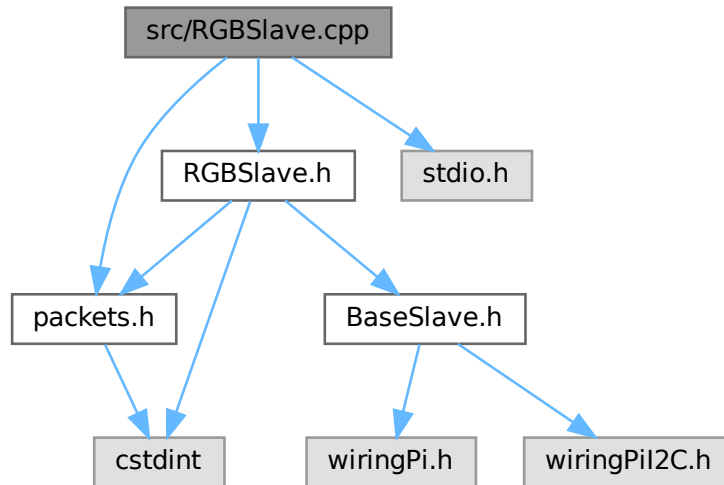
```
00001
00006 #include "math.h"
00007
00016 int add(int a, int b) { return a + b; }
00017
00026 int subtract(int a, int b) { return a - b; }
```

7.28 src/RGBSlave.cpp File Reference

```
#include "RGBSlave.h"
#include <stdio.h>
```

```
#include "packets.h"
```

Include dependency graph for RGBSlave.cpp:



7.29 RGBSlave.cpp

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

```

00001 #include "RGBSlave.h"
00002
00003 #include <stdio.h>
00004
00005 #include "packets.h"
00006
00007 RGBSlave::RGBSlave(uint8_t id, uint8_t i2c_address) : id(id), i2c_address(i2c_address) {
00008     state_packet.header.ptype = PacketType::DATA;
00009     state_packet.header.length = sizeof(struct sensor_packet_rgb_light);
00010
00011     state_packet.data.rgb_light.metadata.sensor_id = id;
00012     state_packet.data.rgb_light.metadata.sensor_type = SensorType::RGB_LIGHT;
00013 }
00014
00015 const void* RGBSlave::getData() {
00016     uint8_t r, g, b;
00017
00018     r = wiringPiI2CRead(fd);
00019     g = wiringPiI2CRead(fd);
00020     b = wiringPiI2CRead(fd);
00021
00022     color_state.R = r;
00023     color_state.G = g;
00024     color_state.B = b;
00025
00026     state_packet.data.rgb_light.red_state = r;
00027     state_packet.data.rgb_light.green_state = g;
00028     state_packet.data.rgb_light.blue_state = b;
00029
00030     return &state_packet;
00031 }
00032
00033 bool RGBSlave::getStatus() {
00034     getData();
00035     if (color_state.R == 0 && color_state.G == 0 && color_state.B == 0) {
00036         return false;
00037     } else {
00038         return true;
00039     }
00040 }

```

```

00040 }
00041
00042 int RGBSlave::getId() { return id; }
00043
00044 void RGBSlave::setData(void* data) {
00045     RGBData* rgb_data = (RGBData*)data;
00046     uint8_t r = rgb_data->R;
00047     uint8_t g = rgb_data->G;
00048     uint8_t b = rgb_data->B;
00049
00050     char command[256] = { 0 };
00051     sprintf(command, sizeof(command) - 1, "/usr/sbin/i2cset -y 1 0x%02x 0x%02x 0x%02x 0x%02x i", id,
r, g, b);
00052
00053     popen(command, "r");
00054
00055     // wiringPiI2CWriteBlockData(fd, 1, (uint8_t*)rgb_data, 3);
00056     // wiringPiI2CWrite(fd, g);
00057     // wiringPiI2CWrite(fd, b);
00058 }
00059
00060 void RGBSlave::start(int i2c_fd) { fd = i2c_fd; }
00061
00062 void RGBSlave::stop() { fd = -1; }

```

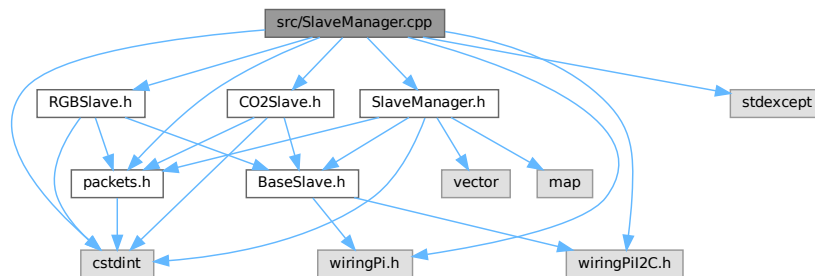
7.30 src/SlaveManager.cpp File Reference

```

#include "SlaveManager.h"
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include <stdint>
#include <stdexcept>
#include "CO2Slave.h"
#include "RGBSlave.h"
#include "packets.h"

```

Include dependency graph for SlaveManager.cpp:



Macros

- `#define MASTER_ADDRESS 0x01`

7.30.1 Macro Definition Documentation

7.30.1.1 MASTER_ADDRESS

```
#define MASTER_ADDRESS 0x01
```

Definition at line 13 of file [SlaveManager.cpp](#).

7.31 SlaveManager.cpp

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

```

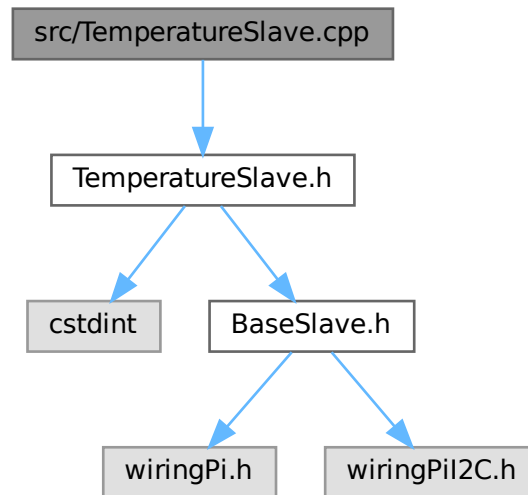
00001 #include "SlaveManager.h"
00002
00003 #include <wiringPi.h>
00004 #include <wiringPiI2C.h>
00005
00006 #include <stdint>
00007 #include <stdexcept>
00008
00009 #include "CO2Slave.h"
00010 #include "RGBSlave.h"
00011 #include "packets.h"
00012
00013 #define MASTER_ADDRESS 0x01
00014
00015 SlaveManager::SlaveManager() : address(MASTER_ADDRESS) {}
00016
00017 void SlaveManager::initialize() {
00018     // maybe need setup not sure.
00019
00020     i2c_fd = wiringPiI2CSetup(address);
00021     if (-1 == i2c_fd) {
00022         throw std::runtime_error("I2C setup failed");
00023     } else {
00024         printf("I2C setup succesful");
00025     };
00026 }
00027
00028 void SlaveManager::ledControl(uint8_t led_number, uint8_t led_state) {}
00029
00030 void SlaveManager::createSlave(SensorType type, int i2c_address) {
00031     BaseSlave* newSlave = nullptr;
00032     switch (type) {
00033         case SensorType::CO2:
00034             newSlave = new CO2Slave(i2c_address, i2c_address);
00035             break;
00036         case SensorType::RGB_LIGHT:
00037             newSlave = new RGBSlave(i2c_address, i2c_address);
00038             break;
00039         // case /* door */:
00040         //     newSlave = new DoorSlave(id, i2c_address);
00041         //     break;
00042         default:
00043             return; // Invalid type
00044     }
00045     slaves[i2c_address] = newSlave;
00046 }
00047 void SlaveManager::deleteSlave(uint8_t id) {
00048     if (nullptr != slaves[id]) {
00049         /*
00050          * Disconnect I2C device from bus
00051          */
00052         delete slaves[id];
00053     }
00054 }
00055
00056 BaseSlave* SlaveManager::getSlave(int id) { return slaves[id]; }

```

7.32 src/TemperatureSlave.cpp File Reference

```
#include "TemperatureSlave.h"
```

Include dependency graph for TemperatureSlave.cpp:



7.33 TemperatureSlave.cpp

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

```

00001 #include "TemperatureSlave.h"
00002
00003 TemperatureSlave::TemperatureSlave(uint8_t id, uint8_t i2c_address)
00004     : id(id), i2c_address(i2c_address) {}
00005
00006 void* TemperatureSlave::getData() {
00007     int temp;
00008
00009     temp = wiringPiI2CRead(fd);
00010
00011     return;
00012 }
00013
00014 bool TemperatureSlave::getStatus() {
00015     getData();
00016     if (temperature.Temp == 0) {
00017         return false;
00018     } else {
00019         return true;
00020     }
00021 }
00022
00023 int TemperatureSlave::getId() { return id; }
00024
00025 void TemperatureSlave::setData(void* data) {
00026     TemperatureData* temp_data = (TemperatureData*)data;
00027     int temp = temp_data->Temp;
00028
00029     wiringPiI2CWrite(fd, temp);
00030 }
00031
00032 void TemperatureSlave::start(int i2c_fd) { fd = i2c_fd; }
00033
00034 void TemperatureSlave::stop() { fd = -1; }

```

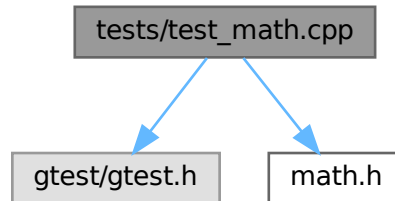
7.34 tests/test_math.cpp File Reference

Unit tests for mathematical operations using Google Test framework.

```
#include <gtest/gtest.h>
```

```
#include "math.h"
```

Include dependency graph for test_math.cpp:



Functions

- [TEST](#) (MathTest, Add)
- [TEST](#) (MathTest, Subtract)
- [TEST](#) (MathTest, SubtractNegative)

7.34.1 Detailed Description

Unit tests for mathematical operations using Google Test framework.

This file contains test cases for verifying the correctness of functions defined in the "math.h" header. The tests ensure that the mathematical operations behave as expected under various conditions.

[Test](#) MathTest.Add

- Verifies that the `add` function correctly computes the sum of two integers.
- Example: `add(2, 3)` should return 5.

[Test](#) MathTest.Subtract

- Verifies that the `subtract` function correctly computes the difference between two integers.
- Examples:
 - `subtract(10, 3)` should return 7.
 - `subtract(9, 3)` should return 6.

[Test](#) MathTest.SubtractNegative

- Verifies that the `subtract` function handles subtraction with negative integers correctly.
- Example: `subtract(10, -3)` should return 13.

Definition in file [test_math.cpp](#).

7.34.2 Function Documentation

7.34.2.1 TEST() [1/3]

```
TEST (
    MathTest ,
    Add )
```

Definition at line 29 of file [test_math.cpp](#).

7.34.2.2 TEST() [2/3]

```
TEST (
    MathTest ,
    Subtract )
```

Definition at line 31 of file [test_math.cpp](#).

7.34.2.3 TEST() [3/3]

```
TEST (
    MathTest ,
    SubtractNegative )
```

Definition at line 36 of file [test_math.cpp](#).

7.35 test_math.cpp

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

```
00001
00025 #include <gtest/gtest.h>
00026
00027 #include "math.h"
00028
00029 TEST(MathTest, Add) { EXPECT_EQ(add(2, 3), 5); }
00030
00031 TEST(MathTest, Subtract) {
00032     EXPECT_EQ(subtract(10, 3), 7);
00033     EXPECT_EQ(subtract(9, 3), 6);
00034 }
00035
00036 TEST(MathTest, SubtractNegative) { EXPECT_EQ(subtract(10, -3), 13); }
```


Index

- `__attribute__`
 - `packets.h`, [64](#)
 - `RGBSlave.h`, [69](#)
 - `TemperatureSlave.h`, [72](#)
 - `~BaseSlave`
 - `BaseSlave`, [12](#)
- `add`
 - `math.cpp`, [80](#)
 - `math.h`, [60](#)
- `address`
 - `SlaveManager`, [48](#)
- B**
 - `RGBData`, [23](#)
 - `RGBSlave.h`, [69](#)
- `BaseSlave`, [11](#)
 - `~BaseSlave`, [12](#)
 - `getData`, [12](#)
 - `getId`, [12](#)
 - `getStatus`, [13](#)
 - `setData`, [13](#)
 - `start`, [13](#)
 - `stop`, [13](#)
- `blue_state`
 - `packets.h`, [64](#)
 - `sensor_packet_rgb_light`, [44](#)
- `buffer`
 - `BusServer`, [16](#)
- `BUFFER_SIZE`
 - `BusServer.h`, [57](#)
- `Bus Bridge Server`, [1](#)
- `BusServer`, [14](#)
 - `buffer`, [16](#)
 - `BusServer`, [15](#)
 - `getSlaveManager`, [15](#)
 - `listen`, [15](#)
 - `listening_address`, [16](#)
 - `listening_fd`, [17](#)
 - `send`, [15](#)
 - `setup`, [16](#)
 - `slave_manager`, [17](#)
 - `start`, [16](#)
 - `wemos_bridge_connected`, [17](#)
- `BusServer.cpp`
 - `RGB_SLAVE_ADDRESS`, [74](#)
- `BusServer.h`
 - `BUFFER_SIZE`, [57](#)
- `BUTTON`
 - `packets.h`, [64](#)
- `CO2`
 - `packets.h`, [64](#)
- `co2`
 - `sensor_data`, [29](#)
 - `sensor_packet::sensor_data`, [30](#)
- `CO2Slave`, [17](#)
 - `CO2Slave`, [20](#)
 - `command`, [21](#)
 - `fd`, [21](#)
 - `getData`, [20](#)
 - `getId`, [20](#)
 - `getStatus`, [20](#)
 - `i2c_address`, [21](#)
 - `id`, [21](#)
 - `power_state`, [21](#)
 - `setData`, [20](#)
 - `start`, [20](#)
 - `state_packet`, [21](#)
 - `stop`, [21](#)
- `color_state`
 - `RGBSlave`, [27](#)
- `command`
 - `CO2Slave`, [21](#)
- `createSlave`
 - `SlaveManager`, [47](#)
- `DASHBOARD_GET`
 - `packets.h`, [63](#)
- `DASHBOARD_POST`
 - `packets.h`, [63](#)
- `DASHBOARD_RESPONSE`
 - `packets.h`, [63](#)
- `DATA`
 - `packets.h`, [63](#)
- `data`
 - `packets.h`, [64](#)
 - `sensor_packet`, [37](#)
- `deleteSlave`
 - `SlaveManager`, [47](#)
- `fd`
 - `CO2Slave`, [21](#)
 - `RGBSlave`, [27](#)
 - `TemperatureSlave`, [54](#)
- G**
 - `RGBData`, [23](#)
 - `RGBSlave.h`, [69](#)
- `generic`
 - `sensor_data`, [29](#)

- sensor_packet::sensor_data, 30
- getData
 - BaseSlave, 12
 - CO2Slave, 20
 - RGBSlave, 26
 - TemperatureSlave, 53
- getId
 - BaseSlave, 12
 - CO2Slave, 20
 - RGBSlave, 26
 - TemperatureSlave, 53
- getSlave
 - SlaveManager, 48
- getSlaveManager
 - BusServer, 15
- getStatus
 - BaseSlave, 13
 - CO2Slave, 20
 - RGBSlave, 26
 - TemperatureSlave, 53
- green_state
 - packets.h, 64
 - sensor_packet_rgb_light, 44
- header
 - packets.h, 64
 - sensor_packet, 37
- HEARTBEAT
 - packets.h, 63
- heartbeat
 - sensor_data, 29
 - sensor_packet::sensor_data, 30
- HUMIDITY
 - packets.h, 64
- humidity
 - sensor_data, 29
 - sensor_packet::sensor_data, 31
- i2c_address
 - CO2Slave, 21
 - RGBSlave, 27
 - TemperatureSlave, 54
- i2c_fd
 - SlaveManager, 48
- id
 - CO2Slave, 21
 - RGBSlave, 27
 - TemperatureSlave, 54
- include/BaseSlave.h, 55, 56
- include/BusServer.h, 56, 57
- include/CO2Slave.h, 58
- include/math.h, 59, 60
- include/packets.h, 61, 66
- include/RGBSlave.h, 67, 69
- include/SlaveManager.h, 70
- include/TemperatureSlave.h, 71, 73
- initialize
 - SlaveManager, 48
- initialized
 - SlaveManager, 49
- ledControl
 - SlaveManager, 48
- length
 - packets.h, 65
 - sensor_header, 32
- LIGHT
 - packets.h, 64
- light
 - sensor_data, 29
 - sensor_packet::sensor_data, 31
- listen
 - BusServer, 15
- listening_address
 - BusServer, 16
- listening_fd
 - BusServer, 17
- main
 - main.cpp, 78
- main.cpp
 - main, 78
- MASTER_ADDRESS
 - SlaveManager.cpp, 83
- math.cpp
 - add, 80
 - subtract, 81
- math.h
 - add, 60
 - subtract, 60
- metadata
 - packets.h, 65
 - sensor_heartbeat, 34
 - sensor_packet_co2, 38
 - sensor_packet_generic, 40
 - sensor_packet_humidity, 41
 - sensor_packet_light, 42
 - sensor_packet_rgb_light, 44
 - sensor_packet_temperature, 46
- MOTION
 - packets.h, 64
- NOOP
 - packets.h, 64
- packets.h
 - __attribute__, 64
 - blue_state, 64
 - BUTTON, 64
 - CO2, 64
 - DASHBOARD_GET, 63
 - DASHBOARD_POST, 63
 - DASHBOARD_RESPONSE, 63
 - DATA, 63
 - data, 64
 - green_state, 64
 - header, 64
 - HEARTBEAT, 63

- HUMIDITY, [64](#)
- length, [65](#)
- LIGHT, [64](#)
- metadata, [65](#)
- MOTION, [64](#)
- NOOP, [64](#)
- PacketType, [63](#)
- PRESSURE, [64](#)
- ptype, [65](#)
- red_state, [65](#)
- RGB_LIGHT, [64](#)
- sensor_id, [65](#)
- sensor_type, [65](#)
- SensorType, [63](#)
- target_state, [66](#)
- TEMPERATURE, [64](#)
- value, [66](#)
- PacketType
 - packets.h, [63](#)
- power_state
 - CO2Slave, [21](#)
 - TemperatureSlave, [54](#)
- PRESSURE
 - packets.h, [64](#)
- ptype
 - packets.h, [65](#)
 - sensor_header, [32](#)
- R
 - RGBData, [23](#)
 - RGBSlave.h, [69](#)
- README.md, [73](#)
- red_state
 - packets.h, [65](#)
 - sensor_packet_rgb_light, [44](#)
- RGB_LIGHT
 - packets.h, [64](#)
- rgb_light
 - sensor_data, [29](#)
 - sensor_packet::sensor_data, [31](#)
- RGB_SLAVE_ADDRESS
 - BusServer.cpp, [74](#)
- RGBData, [22](#)
 - B, [23](#)
 - G, [23](#)
 - R, [23](#)
- RGBSlave, [23](#)
 - color_state, [27](#)
 - fd, [27](#)
 - getData, [26](#)
 - getId, [26](#)
 - getStatus, [26](#)
 - i2c_address, [27](#)
 - id, [27](#)
 - RGBSlave, [26](#)
 - setData, [26](#)
 - start, [26](#)
 - state_packet, [27](#)
 - stop, [27](#)
- RGBSlave.h
 - __attribute__, [69](#)
 - B, [69](#)
 - G, [69](#)
 - R, [69](#)
- send
 - BusServer, [15](#)
- sensor_data, [28](#)
 - co2, [29](#)
 - generic, [29](#)
 - heartbeat, [29](#)
 - humidity, [29](#)
 - light, [29](#)
 - rgb_light, [29](#)
 - temperature, [29](#)
- sensor_header, [32](#)
 - length, [32](#)
 - ptype, [32](#)
- sensor_heartbeat, [33](#)
 - metadata, [34](#)
- sensor_id
 - packets.h, [65](#)
 - sensor_metadata, [35](#)
- sensor_metadata, [34](#)
 - sensor_id, [35](#)
 - sensor_type, [35](#)
- sensor_packet, [35](#)
 - data, [37](#)
 - header, [37](#)
- sensor_packet::sensor_data, [30](#)
 - co2, [30](#)
 - generic, [30](#)
 - heartbeat, [30](#)
 - humidity, [31](#)
 - light, [31](#)
 - rgb_light, [31](#)
 - temperature, [31](#)
- sensor_packet_co2, [37](#)
 - metadata, [38](#)
 - value, [38](#)
- sensor_packet_generic, [39](#)
 - metadata, [40](#)
- sensor_packet_humidity, [40](#)
 - metadata, [41](#)
 - value, [41](#)
- sensor_packet_light, [41](#)
 - metadata, [42](#)
 - target_state, [42](#)
- sensor_packet_rgb_light, [43](#)
 - blue_state, [44](#)
 - green_state, [44](#)
 - metadata, [44](#)
 - red_state, [44](#)
- sensor_packet_temperature, [45](#)
 - metadata, [46](#)
 - value, [46](#)
- sensor_type
 - packets.h, [65](#)

- sensor_metadata, 35
- SensorType
 - packets.h, 63
- setData
 - BaseSlave, 13
 - CO2Slave, 20
 - RGBSlave, 26
 - TemperatureSlave, 53
- setup
 - BusServer, 16
- slave_manager
 - BusServer, 17
- SlaveManager, 46
 - address, 48
 - createSlave, 47
 - deleteSlave, 47
 - getSlave, 48
 - i2c_fd, 48
 - initialize, 48
 - initialized, 49
 - ledControl, 48
 - SlaveManager, 47
 - slaves, 49
- SlaveManager.cpp
 - MASTER_ADDRESS, 83
- slaves
 - SlaveManager, 49
- src/BaseSlave.cpp, 73
- src/BusServer.cpp, 74
- src/CO2Slave.cpp, 77
- src/main.cpp, 78
- src/math.cpp, 79, 81
- src/RGBSlave.cpp, 81, 82
- src/SlaveManager.cpp, 83, 84
- src/TemperatureSlave.cpp, 85
- start
 - BaseSlave, 13
 - BusServer, 16
 - CO2Slave, 20
 - RGBSlave, 26
 - TemperatureSlave, 53
- state_packet
 - CO2Slave, 21
 - RGBSlave, 27
- stop
 - BaseSlave, 13
 - CO2Slave, 21
 - RGBSlave, 27
 - TemperatureSlave, 54
- subtract
 - math.cpp, 81
 - math.h, 60
- target_state
 - packets.h, 66
 - sensor_packet_light, 42
- Temp
 - TemperatureData, 50
 - TemperatureSlave.h, 72

- TEMPERATURE
 - packets.h, 64
- temperature
 - sensor_data, 29
 - sensor_packet::sensor_data, 31
 - TemperatureSlave, 54
- TemperatureData, 49
 - Temp, 50
- TemperatureSlave, 50
 - fd, 54
 - getData, 53
 - getId, 53
 - getStatus, 53
 - i2c_address, 54
 - id, 54
 - power_state, 54
 - setData, 53
 - start, 53
 - stop, 54
 - temperature, 54
 - TemperatureSlave, 53
- TemperatureSlave.h
 - __attribute__, 72
 - Temp, 72
- TEST
 - test_math.cpp, 87
- Test List, 3
- test_math.cpp
 - TEST, 87
- tests/test_math.cpp, 86, 87
- value
 - packets.h, 66
 - sensor_packet_co2, 38
 - sensor_packet_humidity, 41
 - sensor_packet_temperature, 46
- wemos_bridge_connected
 - BusServer, 17