

Bus Bridge Server

commit-4547d98

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	13
6.1.2 Constructor & Destructor Documentation	13
6.1.2.1 ~BaseSlave()	13
6.1.3 Member Function Documentation	13
6.1.3.1 getData()	13
6.1.3.2 getId()	14
6.1.3.3 getStatus()	14
6.1.3.4 setData()	14
6.1.3.5 start()	14
6.1.3.6 stop()	14
6.2 BusServer Class Reference	15
6.2.1 Detailed Description	15
6.2.2 Constructor & Destructor Documentation	16
6.2.2.1 BusServer()	16
6.2.3 Member Function Documentation	16
6.2.3.1 listen()	16
6.2.3.2 send()	16
6.2.3.3 setup()	16
6.2.3.4 start()	17
6.2.4 Member Data Documentation	17
6.2.4.1 buffer	17
6.2.4.2 listening_address	17
6.2.4.3 listening_fd	17
6.2.4.4 slave_manager	18
6.2.4.5 wemos_bridge_connected	18
6.3 CO2Slave Class Reference	18
6.3.1 Detailed Description	20
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 fd	21
6.3.4.2 id	21
6.4 RGBData Struct Reference	22
6.4.1 Detailed Description	22
6.4.2 Member Data Documentation	22
6.4.2.1 B	22
6.4.2.2 G	22
6.4.2.3 R	23
6.5 RGBSlave Class Reference	23
6.5.1 Detailed Description	25
6.5.2 Constructor & Destructor Documentation	25
6.5.2.1 RGBSlave()	25
6.5.3 Member Function Documentation	25
6.5.3.1 getData()	25
6.5.3.2 getId()	25
6.5.3.3 getStatus()	25
6.5.3.4 setData()	26
6.5.3.5 start()	26
6.5.3.6 stop()	26
6.5.4 Member Data Documentation	26
6.5.4.1 color_state	26
6.5.4.2 fd	26
6.5.4.3 i2c_address	26
6.5.4.4 id	27
6.5.4.5 power_state	27
6.6 sensor_data Union Reference	27
6.6.1 Detailed Description	28
6.6.2 Member Data Documentation	28
6.6.2.1 co2	28
6.6.2.2 generic	28
6.6.2.3 heartbeat	28
6.6.2.4 humidity	28
6.6.2.5 light	28
6.6.2.6 rgb_light	28

6.6.2.7 temperature	29
6.7 sensor_packet::sensor_data Union Reference	29
6.7.1 Detailed Description	29
6.7.2 Member Data Documentation	30
6.7.2.1 co2	30
6.7.2.2 generic	30
6.7.2.3 heartbeat	30
6.7.2.4 humidity	30
6.7.2.5 light	30
6.7.2.6 rgb_light	30
6.7.2.7 temperature	30
6.8 sensor_header Struct Reference	31
6.8.1 Detailed Description	31
6.8.2 Member Data Documentation	31
6.8.2.1 length	31
6.8.2.2 ptype	32
6.9 sensor_heartbeat Struct Reference	32
6.9.1 Detailed Description	33
6.9.2 Member Data Documentation	33
6.9.2.1 metadata	33
6.10 sensor_metadata Struct Reference	33
6.10.1 Detailed Description	34
6.10.2 Member Data Documentation	34
6.10.2.1 sensor_id	34
6.10.2.2 sensor_type	34
6.11 sensor_packet Struct Reference	34
6.11.1 Detailed Description	35
6.11.2 Member Data Documentation	36
6.11.2.1 data	36
6.11.2.2 header	36
6.12 sensor_packet_co2 Struct Reference	36
6.12.1 Detailed Description	37
6.12.2 Member Data Documentation	37
6.12.2.1 metadata	37
6.12.2.2 value	37
6.13 sensor_packet_generic Struct Reference	38
6.13.1 Detailed Description	38
6.13.2 Member Data Documentation	39
6.13.2.1 metadata	39
6.14 sensor_packet_humidity Struct Reference	39
6.14.1 Detailed Description	40
6.14.2 Member Data Documentation	40

6.14.2.1 metadata	40
6.14.2.2 value	40
6.15 sensor_packet_light Struct Reference	40
6.15.1 Detailed Description	41
6.15.2 Member Data Documentation	41
6.15.2.1 metadata	41
6.15.2.2 target_state	42
6.16 sensor_packet_rgb_light Struct Reference	42
6.16.1 Detailed Description	43
6.16.2 Member Data Documentation	43
6.16.2.1 blue_state	43
6.16.2.2 green_state	43
6.16.2.3 metadata	43
6.16.2.4 red_state	43
6.17 sensor_packet_temperature Struct Reference	44
6.17.1 Detailed Description	44
6.17.2 Member Data Documentation	45
6.17.2.1 metadata	45
6.17.2.2 value	45
6.18 SlaveManager Class Reference	45
6.18.1 Detailed Description	46
6.18.2 Constructor & Destructor Documentation	46
6.18.2.1 SlaveManager()	46
6.18.3 Member Function Documentation	46
6.18.3.1 createSlave()	46
6.18.3.2 deleteSlave()	46
6.18.3.3 getSlave()	47
6.18.3.4 initialize()	47
6.18.3.5 ledControl()	47
6.18.4 Member Data Documentation	47
6.18.4.1 address	47
6.18.4.2 i2c_fd	48
6.18.4.3 initialized	48
6.18.4.4 slaves	48
7 File Documentation	49
7.1 include/BaseSlave.h File Reference	49
7.2 BaseSlave.h	50
7.3 include/BusServer.h File Reference	50
7.3.1 Macro Definition Documentation	51
7.3.1.1 BUFFER_SIZE	51
7.4 BusServer.h	51

7.5 include/CO2Slave.h File Reference	52
7.6 CO2Slave.h	52
7.7 include/math.h File Reference	53
7.7.1 Detailed Description	53
7.7.2 Function Documentation	54
7.7.2.1 add()	54
7.7.2.2 subtract()	54
7.8 math.h	54
7.9 include/packets.h File Reference	55
7.9.1 Detailed Description	57
7.9.2 Enumeration Type Documentation	57
7.9.2.1 PacketType	57
7.9.2.2 SensorType	58
7.9.3 Function Documentation	58
7.9.3.1 __attribute__()	58
7.9.4 Variable Documentation	58
7.9.4.1 blue_state	58
7.9.4.2 data	58
7.9.4.3 green_state	58
7.9.4.4 header	59
7.9.4.5 length	59
7.9.4.6 metadata	59
7.9.4.7 ptype	59
7.9.4.8 red_state	59
7.9.4.9 sensor_id	59
7.9.4.10 sensor_type	60
7.9.4.11 target_state	60
7.9.4.12 value	60
7.10 packets.h	60
7.11 include/RGBSlave.h File Reference	61
7.11.1 Function Documentation	63
7.11.1.1 __attribute__()	63
7.11.2 Variable Documentation	63
7.11.2.1 __attribute__	63
7.11.2.2 B	63
7.11.2.3 G	63
7.11.2.4 R	63
7.12 RGBSlave.h	63
7.13 include/SlaveManager.h File Reference	64
7.14 SlaveManager.h	64
7.15 README.md File Reference	65
7.16 src/BaseSlave.cpp File Reference	65

7.17 BaseSlave.cpp	66
7.18 src/BusServer.cpp File Reference	66
7.18.1 Macro Definition Documentation	66
7.18.1.1 RGB_SLAVE_ADDRESS	66
7.19 BusServer.cpp	67
7.20 src/CO2Slave.cpp File Reference	69
7.21 CO2Slave.cpp	69
7.22 src/main.cpp File Reference	69
7.22.1 Function Documentation	70
7.22.1.1 main()	70
7.23 main.cpp	70
7.24 src/math.cpp File Reference	71
7.24.1 Detailed Description	71
7.24.2 Function Documentation	71
7.24.2.1 add()	71
7.24.2.2 subtract()	72
7.25 math.cpp	72
7.26 src/RGBSlave.cpp File Reference	73
7.27 RGBSlave.cpp	73
7.28 src/SlaveManager.cpp File Reference	74
7.28.1 Macro Definition Documentation	74
7.28.1.1 MASTER_ADDRESS	74
7.29 SlaveManager.cpp	75
7.30 tests/test_math.cpp File Reference	75
7.30.1 Detailed Description	76
7.30.2 Function Documentation	77
7.30.2.1 TEST() [1/3]	77
7.30.2.2 TEST() [2/3]	77
7.30.2.3 TEST() [3/3]	77
7.31 test_math.cpp	77

Index	79
--------------	-----------

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	18
RGBSlave	23
BusServer	15
RGBData	22
sensor_data	27
sensor_packet::sensor_data	29
sensor_header	31
sensor_heartbeat	32
sensor_metadata	33
sensor_packet	34
sensor_packet_co2	36
sensor_packet_generic	38
sensor_packet_humidity	39
sensor_packet_light	40
sensor_packet_rgb_light	42
sensor_packet_temperature	44
SlaveManager	45

Chapter 4

Class Index

4.1 Class List

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

BaseSlave	11
BusServer	15
CO2Slave	18
RGBData	22
RGBSlave	23
sensor_data	
27	
sensor_packet::sensor_data	
29	
sensor_header	
Header structure for sensor packets	31
sensor_heartbeat	
Structure for heartbeat packets	32
sensor_metadata	
Structure for sensor metadata, which is always included in any packet	33
sensor_packet	
Union structure for the entire sensor packet	34
sensor_packet_co2	
Structure for CO2 sensor packets	36
sensor_packet_generic	
Structure for generic sensor packets	38
sensor_packet_humidity	
Structure for humidity sensor packets	39
sensor_packet_light	
Structure for light sensor packets	40
sensor_packet_rgb_light	
Structure for RGB light sensor packets	42
sensor_packet_temperature	
Structure for temperature sensor packets	44
SlaveManager	45

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/ BaseSlave.h	49
include/ BusServer.h	50
include/ CO2Slave.h	52
include/ math.h	
Header file for math.cpp	53
include/ packets.h	
Header file for packets.h	55
include/ RGBSlave.h	61
include/ SlaveManager.h	64
src/ BaseSlave.cpp	65
src/ BusServer.cpp	66
src/ CO2Slave.cpp	69
src/ main.cpp	69
src/ math.cpp	
Implementation of basic math operations	71
src/ RGBSlave.cpp	73
src/ SlaveManager.cpp	74
tests/ test_math.cpp	
Unit tests for mathematical operations using Google Test framework	75

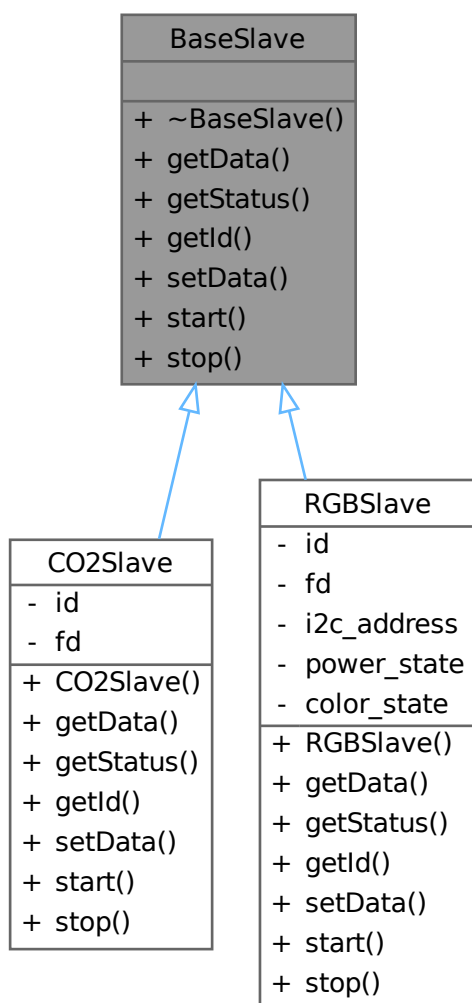
Chapter 6

Class Documentation

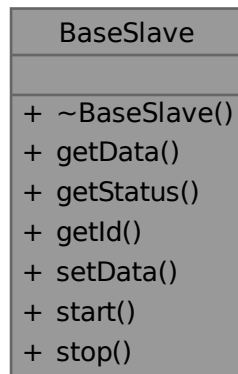
6.1 BaseSlave Class Reference

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

Inheritance diagram for BaseSlave:



Collaboration diagram for BaseSlave:



Public Member Functions

- virtual [~BaseSlave](#) ()=default
- virtual 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 void * BaseSlave::getData ( ) [pure virtual]
```

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

6.1.3.2 getId()

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

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

6.1.3.3 getStatus()

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

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

6.1.3.4 setData()

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

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

6.1.3.5 start()

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

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

6.1.3.6 stop()

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

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

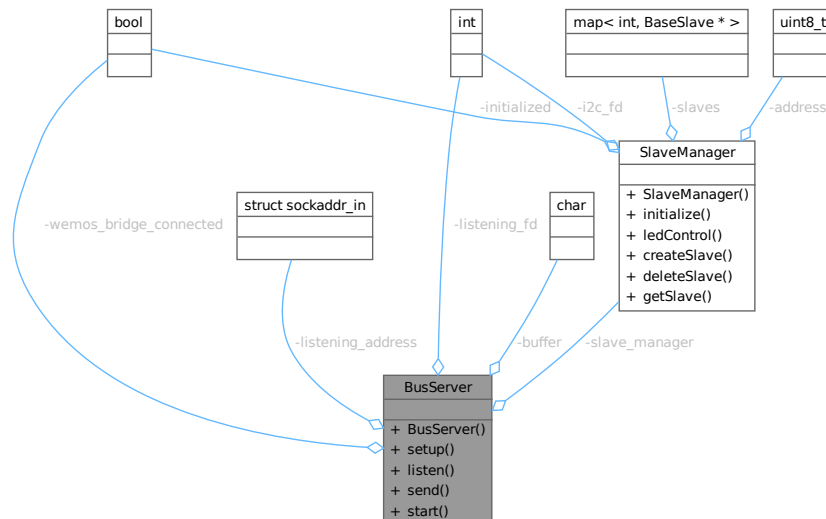
The documentation for this class was generated from the following file:

- [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`.

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 listen()

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

Open the underlying socket for incoming connections.

Exceptions

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

Definition at line 57 of file [BusServer.cpp](#).

6.2.3.2 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 64 of file [BusServer.cpp](#).

6.2.3.3 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

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

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

6.2.3.4 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 74 of file [BusServer.cpp](#).

6.2.4 Member Data Documentation

6.2.4.1 buffer

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

Definition at line 55 of file [BusServer.h](#).

6.2.4.2 listening_address

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

Definition at line 53 of file [BusServer.h](#).

6.2.4.3 listening_fd

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

Definition at line 51 of file [BusServer.h](#).

6.2.4.4 slave_manager

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

Definition at line 57 of file [BusServer.h](#).

6.2.4.5 wemos_bridge_connected

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

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

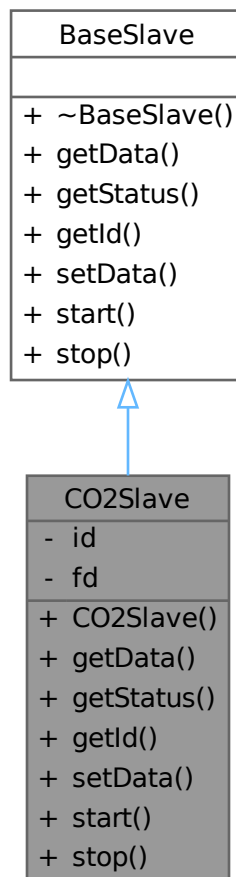
The documentation for this class was generated from the following files:

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

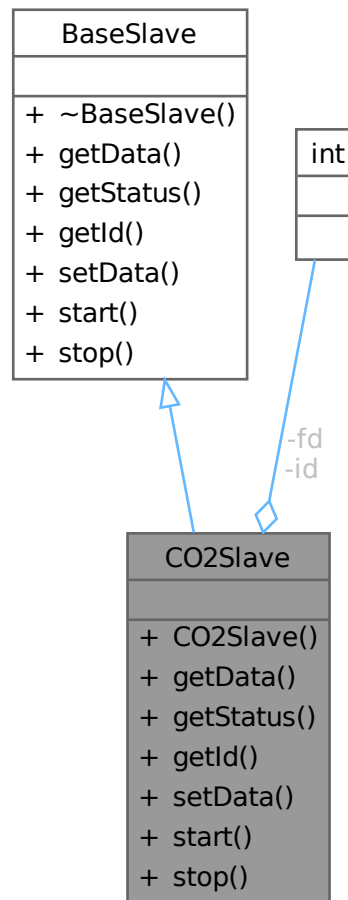
6.3 CO2Slave Class Reference

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

Inheritance diagram for CO2Slave:



Collaboration diagram for CO2Slave:



Public Member Functions

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

Public Member Functions inherited from [BaseSlave](#)

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

Private Attributes

- int [id](#)
- int [fd](#)

6.3.1 Detailed Description

Definition at line 6 of file [CO2Slave.h](#).

6.3.2 Constructor & Destructor Documentation

6.3.2.1 CO2Slave()

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

Definition at line 3 of file [CO2Slave.cpp](#).

6.3.3 Member Function Documentation

6.3.3.1 getData()

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

Implements [BaseSlave](#).

Definition at line 9 of file [CO2Slave.cpp](#).

6.3.3.2 getId()

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

Implements [BaseSlave](#).

Definition at line 13 of file [CO2Slave.cpp](#).

6.3.3.3 getStatus()

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

Implements [BaseSlave](#).

Definition at line 11 of file [CO2Slave.cpp](#).

6.3.3.4 setData()

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

Implements [BaseSlave](#).

Definition at line 15 of file [CO2Slave.cpp](#).

6.3.3.5 start()

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

Implements [BaseSlave](#).

Definition at line 17 of file [CO2Slave.cpp](#).

6.3.3.6 stop()

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

Implements [BaseSlave](#).

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

6.3.4 Member Data Documentation

6.3.4.1 fd

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

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

6.3.4.2 id

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

Definition at line 18 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 6 of file [RGBSlave.h](#).

6.4.2 Member Data Documentation

6.4.2.1 B

```
uint8_t RGBData::B
```

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

6.4.2.2 G

```
uint8_t RGBData::G
```

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

6.4.2.3 R

```
uint8_t RGBData::R
```

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

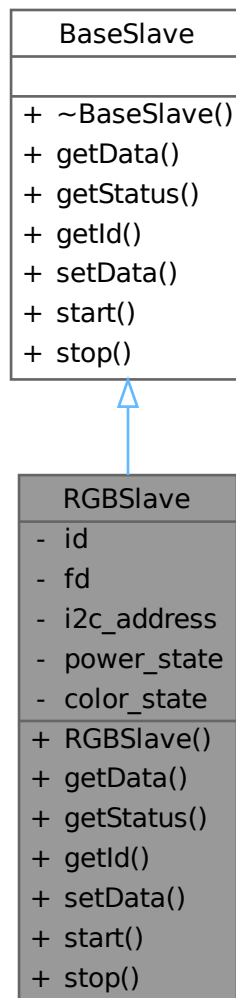
The documentation for this struct was generated from the following file:

- [include/RGBSlave.h](#)

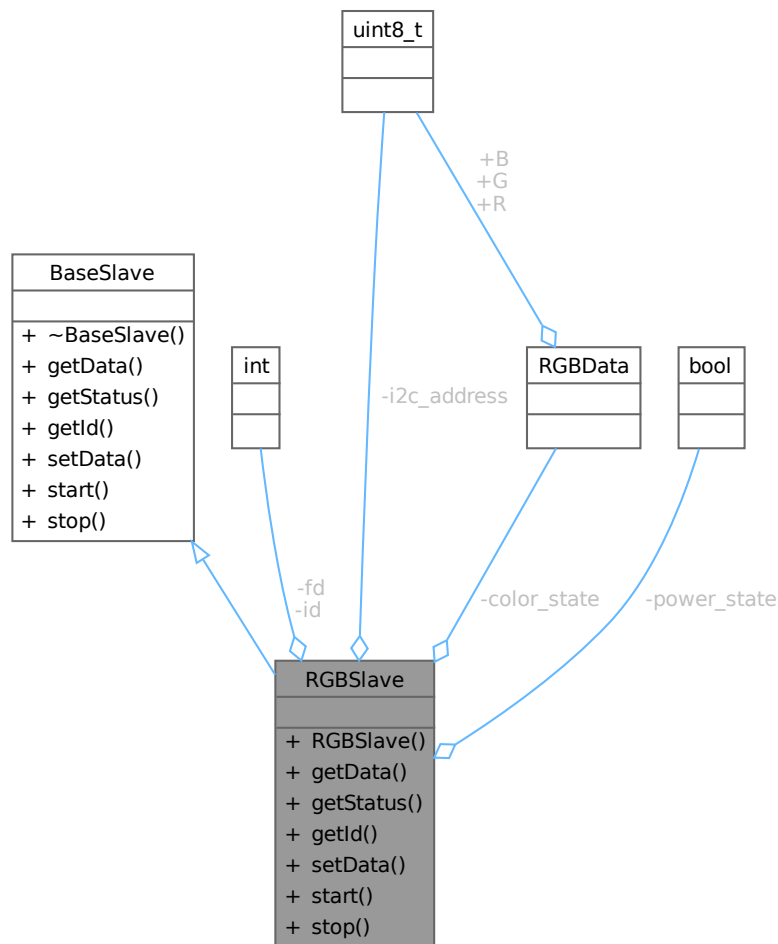
6.5 RGBSlave Class Reference

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

Inheritance diagram for RGBSlave:



Collaboration diagram for RGBSlave:



Public Member Functions

- **RGBSlave** (uint8_t id, uint8_t i2c_address)
- 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](#)
- [RGBData](#) [color_state](#)

6.5.1 Detailed Description

Definition at line 10 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 3 of file [RGBSlave.cpp](#).

6.5.3 Member Function Documentation

6.5.3.1 getData()

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

Implements [BaseSlave](#).

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

6.5.3.2 getId()

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

Implements [BaseSlave](#).

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

6.5.3.3 getStatus()

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

Implements [BaseSlave](#).

Definition at line 19 of file [RGBSlave.cpp](#).

6.5.3.4 setData()

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

Implements [BaseSlave](#).

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

6.5.3.5 start()

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

Implements [BaseSlave](#).

Definition at line 41 of file [RGBSlave.cpp](#).

6.5.3.6 stop()

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

Implements [BaseSlave](#).

Definition at line 43 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 22 of file [RGBSlave.h](#).

6.5.4.3 i2c_address

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

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

6.5.4.4 id

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

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

6.5.4.5 power_state

```
bool RGBSlave::power_state [private]
```

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

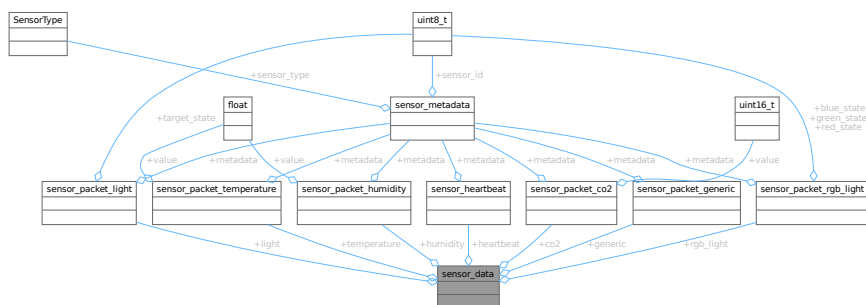
The documentation for this class was generated from the following files:

- 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\_packet\_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](#).

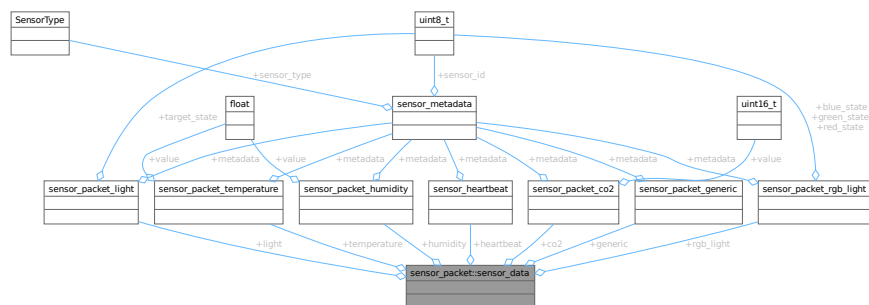
The documentation for this union was generated from the following file:

- [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:

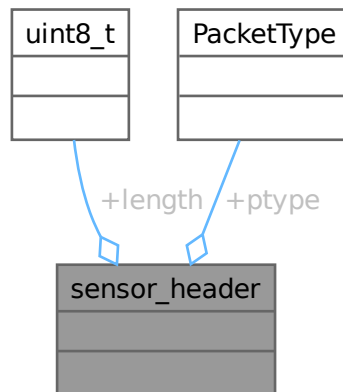
- [include/packets.h](#)

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:

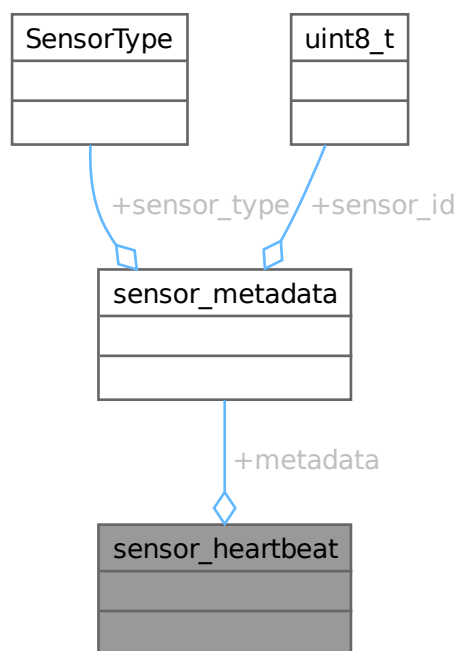
- [include/packets.h](#)

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:

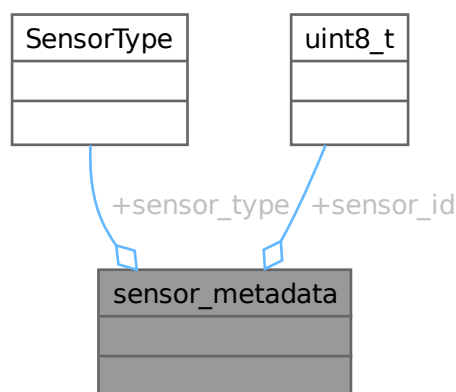
- [include/packets.h](#)

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:

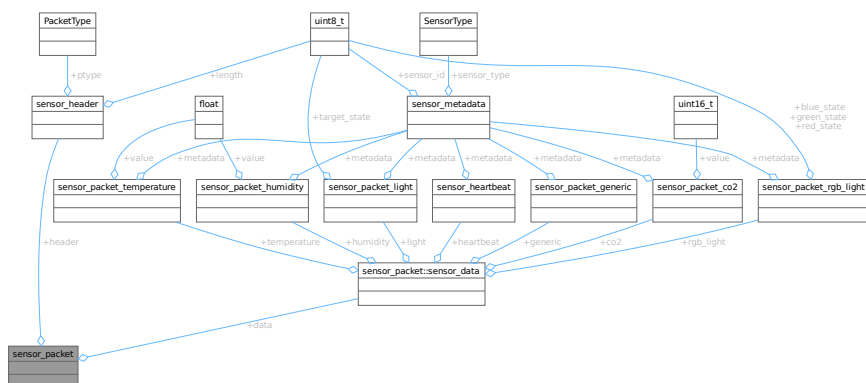
- [include/packets.h](#)

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:

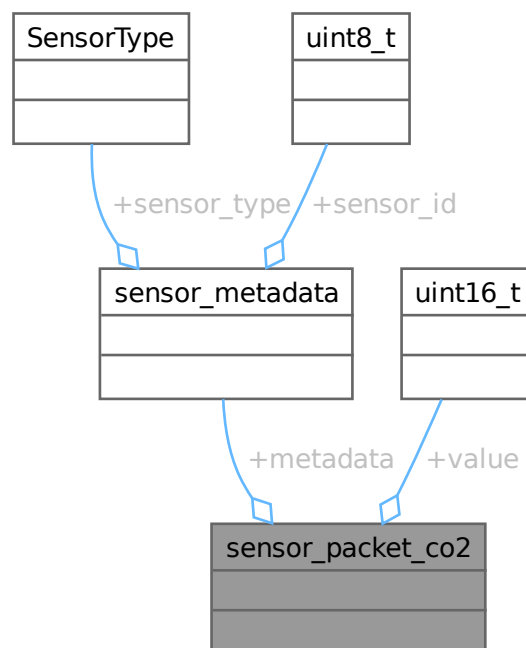
- [include/packets.h](#)

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:

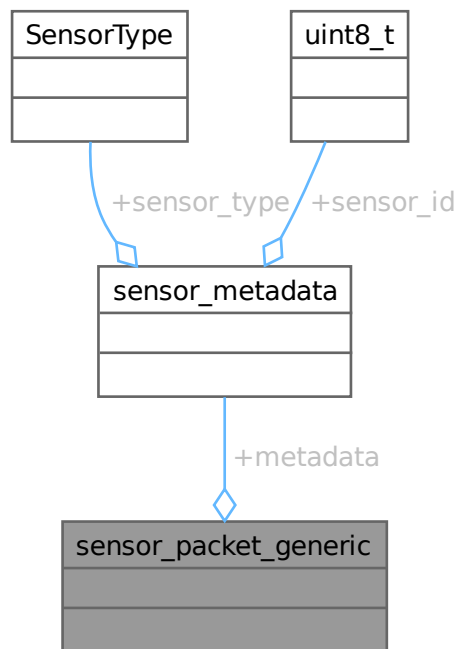
- [include/packets.h](#)

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:

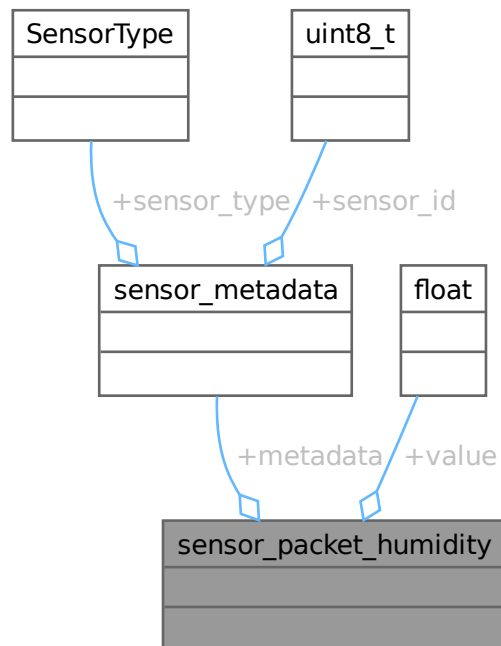
- [include/packets.h](#)

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:

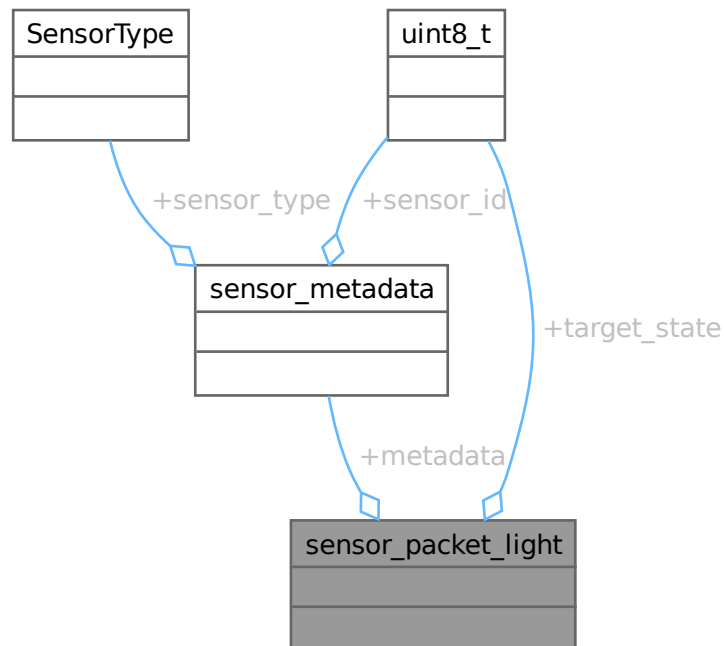
- [include/packets.h](#)

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:

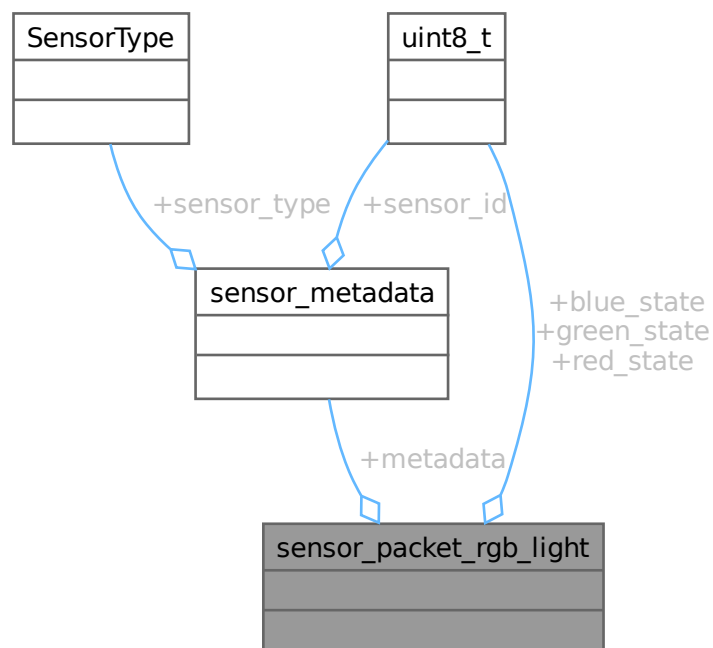
- [include/packets.h](#)

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:

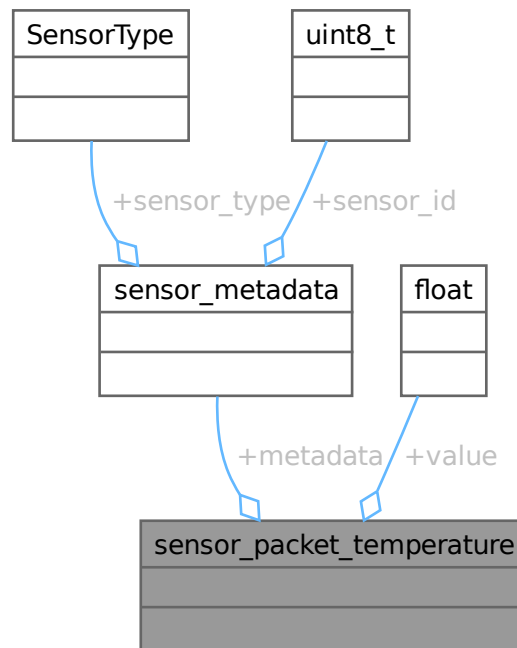
- [include/packets.h](#)

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](#).

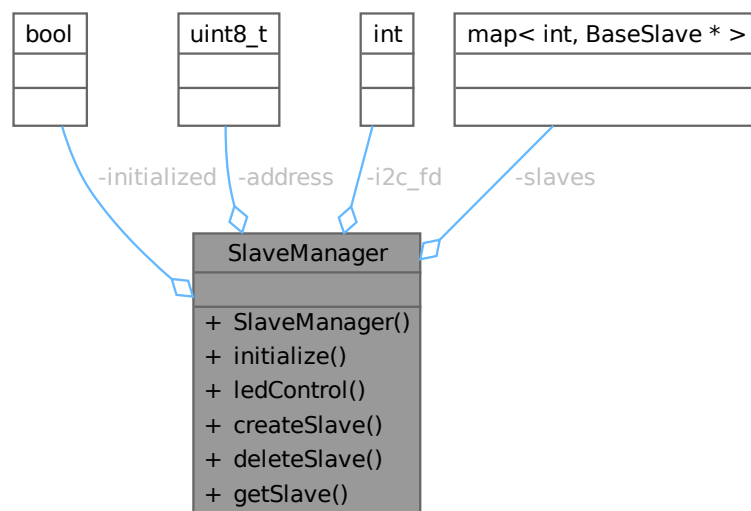
The documentation for this struct was generated from the following file:

- [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, uint8_t id, 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 16 of file [SlaveManager.cpp](#).

6.18.3 Member Function Documentation

6.18.3.1 createSlave()

```
void SlaveManager::createSlave (
    SensorType type,
    uint8_t id,
    int i2c_address )
```

Definition at line 31 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 48 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 57 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 18 of file [SlaveManager.cpp](#).

6.18.3.5 ledControl()

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

Definition at line 29 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](#)

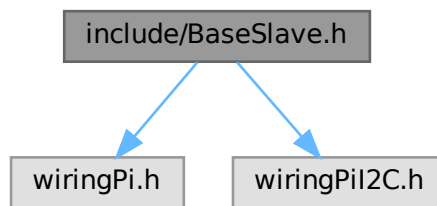
Chapter 7

File Documentation

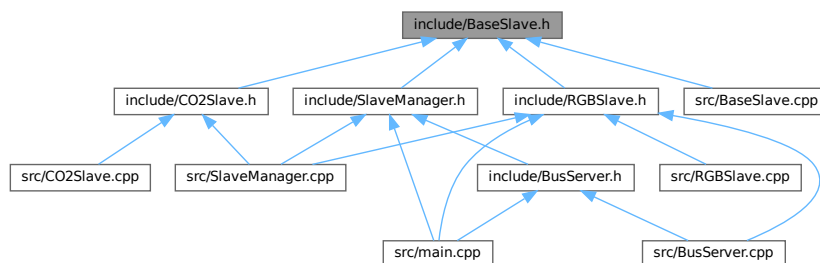
7.1 include/BaseSlave.h File Reference

```
#include <wiringPi.h>
#include <wiringPiI2C.h>
```

Include dependency graph for BaseSlave.h:



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



Classes

- class [BaseSlave](#)

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 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 };

```

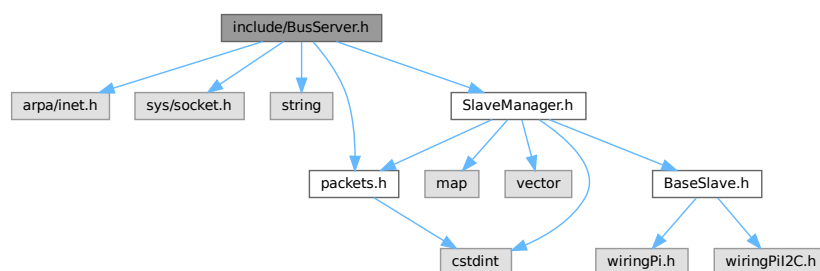
7.3 include/BusServer.h File Reference

```

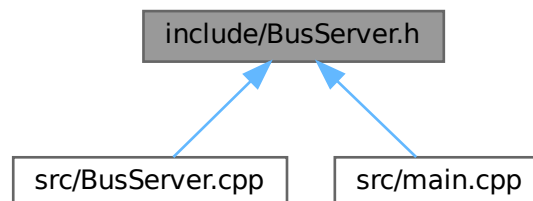
#include <arpa/inet.h>
#include <sys/socket.h>
#include <string>
#include "SlaveManager.h"
#include "packets.h"

```

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](#).

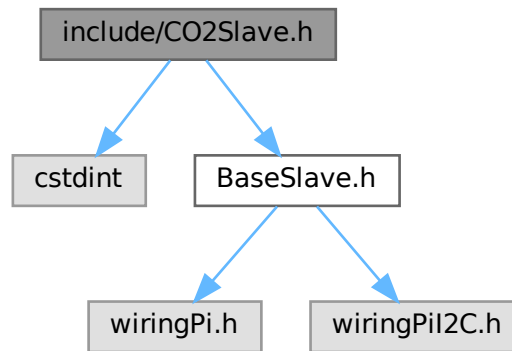
7.4 BusServer.h

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

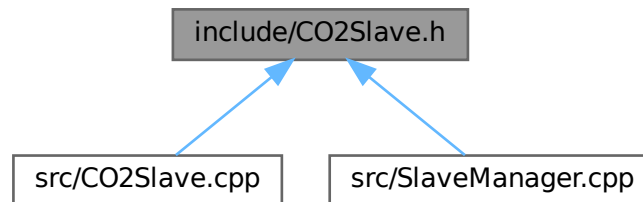
```
00001 #pragma once
00002 #include <arpa/inet.h>
00003 #include <sys/socket.h>
00004
00005 #include <string>
00006
00007 #include "SlaveManager.h"
00008 #include "packets.h"
00009
00010 #define BUFFER_SIZE 1024
00011
00012 class BusServer {
00013     public:
00014         BusServer() : listening_fd(-1) {}
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
00050     private:
00051         int listening_fd;
00052
00053         struct sockaddr_in listening_address;
00054         bool wemos_bridge_connected = false;
00055         char buffer[BUFFER_SIZE];
00056
00057         SlaveManager slave_manager;
00058 };
```

7.5 include/CO2Slave.h File Reference

```
#include <stdint>
#include "BaseSlave.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.](#)

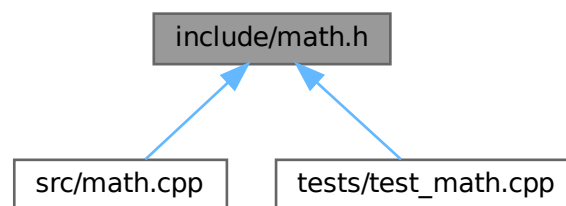
```
00001 #pragma once
00002 #include <stdint>
00003
00004 #include "BaseSlave.h"
00005
```

```
00006 class CO2Slave : public BaseSlave {
00007     public:
00008         CO2Slave(uint8_t id, int i2c_address);
00009
00010         void* getData() override;
00011         bool getStatus() override;
00012         int getId() override;
00013         void setData(void* data) override;
00014         void start(int i2c_fd) override;
00015         void stop() override;
00016
00017     private:
00018         int id;
00019         int fd;
00020 };
```

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

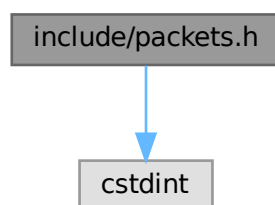
```
00008 #ifndef MATH_H
00009 #define MATH_H
00010
00011 int add(int a, int b);
00012 int subtract(int a, int b);
00013
00014 #endif
```

7.9 include/packets.h File Reference

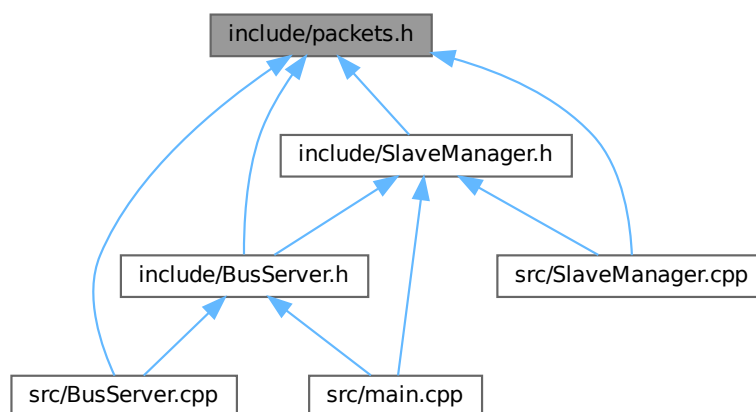
Header file for [packets.h](#).

```
#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

```

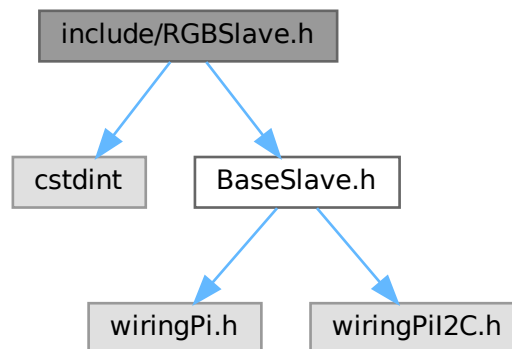
7.11 include/RGBSlave.h File Reference

```

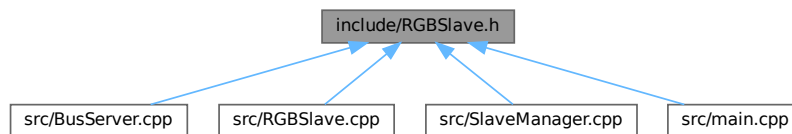
#include <stdint>
#include "BaseSlave.h"

```

Include 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__((packed))

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

7.11.2 Variable Documentation

7.11.2.1 __attribute__((packed))

```
struct sensor_packet __attribute__((packed)) {
```

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

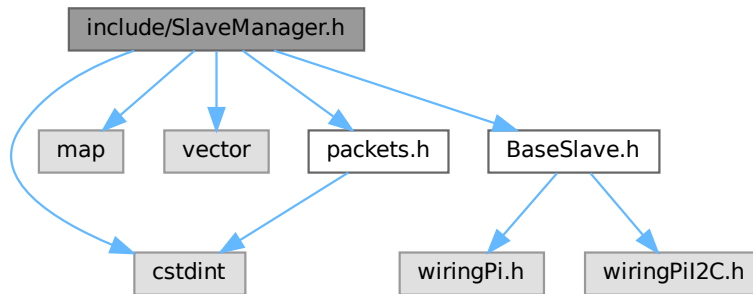
[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <stdint>
00003
00004 #include "BaseSlave.h"
00005
00006 struct RGBData {
00007     uint8_t R, G, B;
00008 } __attribute__((packed));
00009
00010 class RGBSlave : public BaseSlave {
00011     public:
00012         RGBSlave(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     private:
00020         int id;
00021         int fd;
00022         uint8_t i2c_address;
00023
00024         bool power_state;
00025         RGBData color_state;
00026 };
```

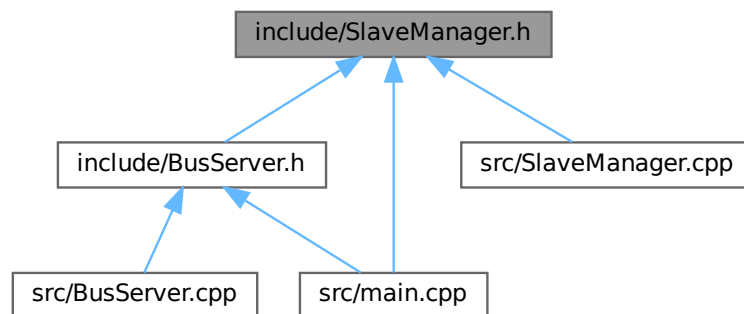
7.13 include/SlaveManager.h File Reference

```
#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 <stdint>
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, uint8_t id, 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 };

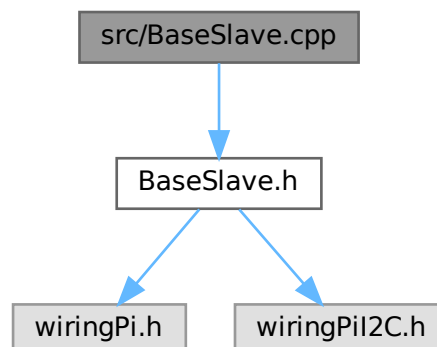
```

7.15 README.md File Reference

7.16 src/BaseSlave.cpp File Reference

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

Include dependency graph for BaseSlave.cpp:



7.17 BaseSlave.cpp

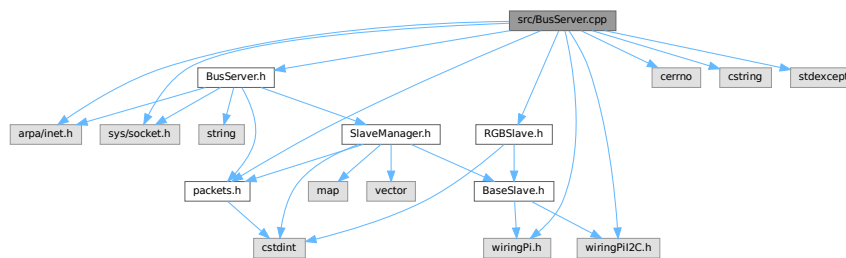
[Go to the documentation of this file.](#)

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

7.18 src/BusServer.cpp File Reference

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

Include dependency graph for BusServer.cpp:



Macros

- `#define RGB_SLAVE_ADDRESS 0x42`

7.18.1 Macro Definition Documentation

7.18.1.1 RGB_SLAVE_ADDRESS

```
#define RGB_SLAVE_ADDRESS 0x42
```

Definition at line 15 of file `BusServer.cpp`.

7.19 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 "RGBSlave.h"
00011 #include "packets.h"
00012 #include "wiringPi.h"
00013 #include "wiringPiI2C.h"
00014
00015 #define RGB_SLAVE_ADDRESS 0x42
00016
00017 void BusServer::setup(std::string ip, int port) {
00018     // slave_manager.initialize();
00019
00020     memset(&listening_address, 0, sizeof(listening_address));
00021
00022     if (port <= 0 || port > 65535) {
00023         throw std::invalid_argument("invalid port passed");
00024     }
00025
00026     if (0 == inet_aton(ip.c_str(), &listening_address.sin_addr)) {
00027         perror("inet_aton()");
00028         throw std::invalid_argument("invalid IP address passed");
00029     }
00030
00031     listening_address.sin_family = AF_INET;
00032     listening_address.sin_port = htons(port);
00033
00034     listening_fd = socket(AF_INET, SOCK_STREAM, 0);
00035     if (-1 == listening_fd) {
00036         perror("socket()");
00037         throw std::runtime_error("failed to create socket");
00038     }
00039
00040     {
00041         int one = 1;
00042         if (-1 ==
00043             setsockopt(listening_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &one, sizeof(one))) {
00044             perror("setsockopt()");
00045             throw std::runtime_error("failed to set socket options");
00046         }
00047     }
00048
00049     socklen_t len = sizeof(listening_address);
00050
00051     if (-1 == bind(listening_fd, (struct sockaddr*)&listening_address, len)) {
00052         perror("bind()");
00053         throw std::runtime_error("failed to bind socket");
00054     }
00055 }
00056
00057 void BusServer::listen() {
00058     if (-1 == ::listen(listening_fd, 8)) {
00059         perror("listen()");
00060         throw std::runtime_error("failed to listen on socket");
00061     }
00062 }
00063
00064 void BusServer::send(struct sensor_packet* packet_ptr, int fd) {
00065     if (!wemos_bridge_connected) {
00066         throw std::runtime_error("no longer connected to socket");
00067     }
00068     if (-1 == ::send(fd, packet_ptr, sizeof(struct sensor_header) + packet_ptr->header.length, 0)) {
00069         perror("send()");
00070         throw std::runtime_error("failed to send on socket");
00071     }
00072 }
00073
00074 void BusServer::start() {
00075     // start off by mapping pre-known addresses to ID's
00076     slave_manager.createSlave(SensorType::RGB_LIGHT, 128, RGB_SLAVE_ADDRESS);
00077
00078     listen();
00079
00080     while (true) {
00081         char buffer[32] = {0};
00082         bool net_request = false;

```

```

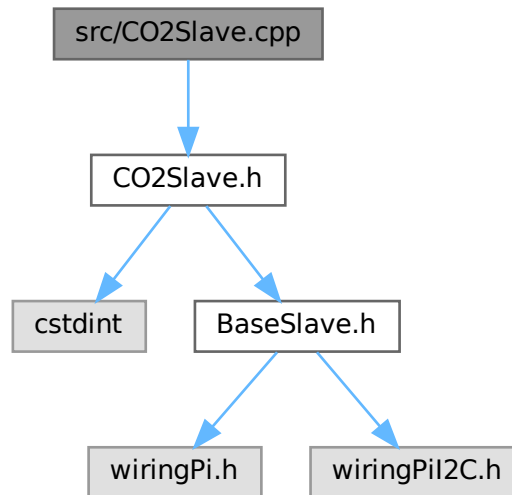
00083
00084 {
00085     struct sockaddr_in client_address = {};
00086     socklen_t client_address_length;
00087
00088     int new_fd = accept4(listening_fd, (struct sockaddr*)&client_address,
00089                         &client_address_length, SOCK_NONBLOCK);
00090
00091     if (-1 == new_fd) {
00092         int err = errno;
00093         switch (err) {
00094             case EWOULDBLOCK:
00095                 // no client has tried to connect; no big deal
00096                 break;
00097
00098             default:
00099                 perror("accept4()");
00100                 throw std::runtime_error("accept4() failed");
00101                 break;
00102         }
00103     } else {
00104         if (-1 == recv(new_fd, buffer, sizeof(buffer), 0)) {
00105             int err = errno;
00106             perror("recv()");
00107             throw std::runtime_error("reading from client socket failed for some reason");
00108         } else {
00109             net_request = true;
00110         }
00111     }
00112 }
00113
00114 if (net_request) {
00115     struct sensor_packet* pkt_ptr = (struct sensor_packet*)buffer;
00116     BaseSlave* the_slave = slave_manager.getSlave(pkt_ptr->data.generic.metadata.sensor_id);
00117     uint8_t values[8] = {0};
00118
00119     switch (pkt_ptr->header.ptype) {
00120         case PacketType::DASHBOARD_POST:
00121             switch (pkt_ptr->data.generic.metadata.sensor_type) {
00122                 case SensorType::LIGHT:
00123                     values[0] = pkt_ptr->data.light.target_state;
00124
00125                     the_slave->setData(values);
00126                     break;
00127
00128                 case SensorType::RGB_LIGHT:
00129                     struct RGBData rgb_data = {.on = 255,
00130                                                .R = pkt_ptr->data.rgb_light.red_state,
00131                                                .G = pkt_ptr->data.rgb_light.green_state,
00132                                                .B = pkt_ptr->data.rgb_light.blue_state};
00133
00134                     the_slave->setData(&rgb_data);
00135                     break;
00136             }
00137             break;
00138         case PacketType::DASHBOARD_GET:
00139             break;
00140     }
00141 }
00142 }
00143 }
00144 }

```

7.20 src/CO2Slave.cpp File Reference

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

Include dependency graph for CO2Slave.cpp:



7.21 CO2Slave.cpp

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

```

00001 #include "CO2Slave.h"
00002
00003 CO2Slave::CO2Slave(uint8_t id, int i2c_address) : id(id) {
00004     /*
00005      * Open I2C connection and save the FD in the 'fd' member variable
00006      */
00007 }
00008
00009 void* CO2Slave::getData() { return nullptr; }
00010
00011 bool CO2Slave::getStatus() { return nullptr; }
00012
00013 int CO2Slave::getId() { return id; }
00014
00015 void CO2Slave::setData(void* data) {}
00016
00017 void CO2Slave::start(int i2c_fd) {}
00018
00019 void CO2Slave::stop() {}

```

7.22 src/main.cpp File Reference

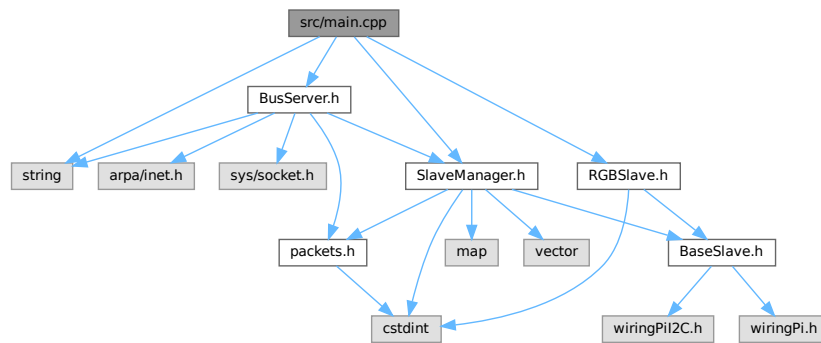
```

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

```

```
#include "SlaveManager.h"
```

Include dependency graph for main.cpp:



Functions

- int [main](#) ()

7.22.1 Function Documentation

7.22.1.1 main()

```
int main ( )
```

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

7.23 main.cpp

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

```

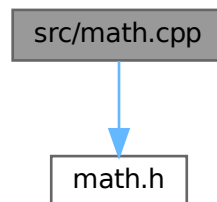
00001 #include <string>
00002
00003 #include "BusServer.h"
00004 #include "RGBSlave.h"
00005 #include "SlaveManager.h"
00006
00007 int main() {
00008     BusServer bus_server;
00009     SlaveManager slave_manager;
00010     sensor_packet_rgb_light packet;
00011     // packet.metadata.sensor_type = SensorType::RGB_LIGHT;
00012     // packet.metadata.sensor_id = 0x80;
00013     packet.red_state = 255;
00014     packet.green_state = 0;
00015     packet.blue_state = 100;
00016
00017     // bus_server.setup("0.0.0.0", 5000);
00018     // bus_server.start();
00019     slave_manager.initialize();
00020
00021     slave_manager.createSlave(SensorType::RGB_LIGHT, 0x80, 0x55);
00022     BaseSlave* base_slave = slave_manager.getSlave(0x80);
00023     RGBSlave* rgb_slave = dynamic_cast<RGBSlave*>(base_slave);
00024     rgb_slave->setData(&packet);
00025 }
```

7.24 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.

7.24.1 Detailed Description

Implementation of basic math operations.

Author

Daan Breur

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

7.24.2 Function Documentation

7.24.2.1 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](#).

7.24.2.2 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](#).

7.25 math.cpp

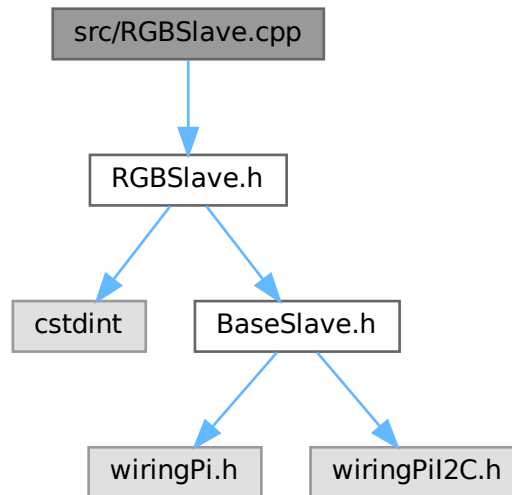
[Go to the documentation of this file.](#)

```
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; }
```


7.26 src/RGBSlave.cpp File Reference

```
#include "RGBSlave.h"
```

Include dependency graph for RGBSlave.cpp:



7.27 RGBSlave.cpp

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

```

00001 #include "RGBSlave.h"
00002
00003 RGBSlave::RGBSlave(uint8_t id, uint8_t i2c_address) : id(id), i2c_address(i2c_address) {}
00004
00005 void* RGBSlave::getData() {
00006     uint8_t r, g, b;
00007
00008     r = wiringPiI2CRead(fd);
00009     g = wiringPiI2CRead(fd);
00010     b = wiringPiI2CRead(fd);
00011
00012     color_state.R = r;
00013     color_state.G = g;
00014     color_state.B = b;
00015
00016     return;
00017 }
00018
00019 bool RGBSlave::getStatus() {
00020     getData();
00021     if (color_state.R == 0 && color_state.G == 0 && color_state.B == 0) {
00022         return false;
00023     } else {
00024         return true;
00025     }
00026 }
00027
00028 int RGBSlave::getId() { return id; }
00029
00030 void RGBSlave::setData(void* data) {
00031     RGBData* rgb_data = (RGBData*)data;
00032     uint8_t r = rgb_data->R;
00033     uint8_t g = rgb_data->G;
00034     uint8_t b = rgb_data->B;

```

```

00035
00036     wiringPiI2CWrite(fd, r);
00037     wiringPiI2CWrite(fd, g);
00038     wiringPiI2CWrite(fd, b);
00039 }
00040
00041 void RGBSlave::start(int i2c_fd) { fd = i2c_fd; }
00042
00043 void RGBSlave::stop() { fd = -1; }

```

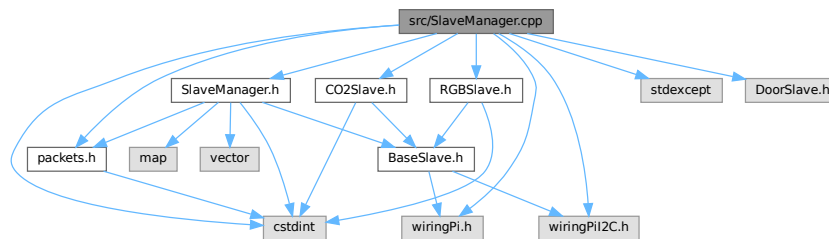
7.28 src/SlaveManager.cpp File Reference

```

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

```

Include dependency graph for SlaveManager.cpp:



Macros

- `#define MASTER_ADDRESS 0x01`

7.28.1 Macro Definition Documentation

7.28.1.1 MASTER_ADDRESS

```
#define MASTER_ADDRESS 0x01
```

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

7.29 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 <cstdint>
00007 #include <stdexcept>
00008
00009 #include "CO2Slave.h"
00010 #include "DoorSlave.h"
00011 #include "RGBSlave.h"
00012 #include "packets.h"
00013
00014 #define MASTER_ADDRESS 0x01
00015
00016 SlaveManager::SlaveManager() : address(MASTER_ADDRESS) {}
00017
00018 void SlaveManager::initialize() {
00019     // maybe need setup not sure.
00020
00021     i2c_fd = wiringPiI2CSetup(address);
00022     if (-1 == i2c_fd) {
00023         throw std::runtime_error("I2C setup failed");
00024     } else {
00025         printf("I2C setup succesful");
00026     };
00027 }
00028
00029 void SlaveManager::ledControl(uint8_t led_number, uint8_t led_state) {}
00030
00031 void SlaveManager::createSlave(SensorType type, uint8_t id, int i2c_address) {
00032     BaseSlave* newSlave = nullptr;
00033     switch (type) {
00034         case SensorType::CO2:
00035             newSlave = new CO2Slave(id, i2c_address);
00036             break;
00037         case SensorType::RGB_LIGHT:
00038             newSlave = new RGBSlave(id, i2c_address);
00039             break;
00040         // case /* door */:
00041         //     newSlave = new DoorSlave(id, i2c_address);
00042         //     break;
00043         default:
00044             return; // Invalid type
00045     }
00046     slaves[id] = newSlave;
00047 }
00048 void SlaveManager::deleteSlave(uint8_t id) {
00049     if (nullptr != slaves[id]) {
00050         /*
00051          * Disconnect I2C device from bus
00052          */
00053         delete slaves[id];
00054     }
00055 }
00056
00057 BaseSlave* SlaveManager::getSlave(int id) { return slaves[id]; }

```

7.30 tests/test_math.cpp File Reference

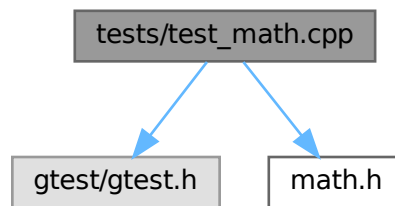
Unit tests for mathematical operations using Google Test framework.

```

#include <gtest/gtest.h>
#include "math.h"

```

Include dependency graph for `test_math.cpp`:



Functions

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

7.30.1 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](#).

7.30.2 Function Documentation

7.30.2.1 TEST() [1/3]

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

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

7.30.2.2 TEST() [2/3]

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

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

7.30.2.3 TEST() [3/3]

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

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

7.31 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`, [58](#)
 - `RGBSlave.h`, [63](#)
 - `~BaseSlave`
 - `BaseSlave`, [13](#)
- `add`
 - `math.cpp`, [71](#)
 - `math.h`, [54](#)
- `address`
 - `SlaveManager`, [47](#)
- B**
 - `RGBData`, [22](#)
 - `RGBSlave.h`, [63](#)
- `BaseSlave`, [11](#)
 - `~BaseSlave`, [13](#)
 - `getData`, [13](#)
 - `getId`, [13](#)
 - `getStatus`, [14](#)
 - `setData`, [14](#)
 - `start`, [14](#)
 - `stop`, [14](#)
- `blue_state`
 - `packets.h`, [58](#)
 - `sensor_packet_rgb_light`, [43](#)
- `buffer`
 - `BusServer`, [17](#)
- `BUFFER_SIZE`
 - `BusServer.h`, [51](#)
- `Bus Bridge Server`, [1](#)
- `BusServer`, [15](#)
 - `buffer`, [17](#)
 - `BusServer`, [16](#)
 - `listen`, [16](#)
 - `listening_address`, [17](#)
 - `listening_fd`, [17](#)
 - `send`, [16](#)
 - `setup`, [16](#)
 - `slave_manager`, [17](#)
 - `start`, [17](#)
 - `wemos_bridge_connected`, [18](#)
- `BusServer.cpp`
 - `RGB_SLAVE_ADDRESS`, [66](#)
- `BusServer.h`
 - `BUFFER_SIZE`, [51](#)
- `BUTTON`
 - `packets.h`, [58](#)
- CO2**
 - `packets.h`, [58](#)
- `co2`
 - `sensor_data`, [28](#)
 - `sensor_packet::sensor_data`, [30](#)
- `CO2Slave`, [18](#)
 - `CO2Slave`, [20](#)
 - `fd`, [21](#)
 - `getData`, [20](#)
 - `getId`, [20](#)
 - `getStatus`, [20](#)
 - `id`, [21](#)
 - `setData`, [20](#)
 - `start`, [20](#)
 - `stop`, [21](#)
- `color_state`
 - `RGBSlave`, [26](#)
- `createSlave`
 - `SlaveManager`, [46](#)
- `DASHBOARD_GET`
 - `packets.h`, [57](#)
- `DASHBOARD_POST`
 - `packets.h`, [57](#)
- `DASHBOARD_RESPONSE`
 - `packets.h`, [57](#)
- `DATA`
 - `packets.h`, [57](#)
- `data`
 - `packets.h`, [58](#)
 - `sensor_packet`, [36](#)
- `deleteSlave`
 - `SlaveManager`, [46](#)
- `fd`
 - `CO2Slave`, [21](#)
 - `RGBSlave`, [26](#)
- G**
 - `RGBData`, [22](#)
 - `RGBSlave.h`, [63](#)
- `generic`
 - `sensor_data`, [28](#)
 - `sensor_packet::sensor_data`, [30](#)
- `getData`
 - `BaseSlave`, [13](#)
 - `CO2Slave`, [20](#)
 - `RGBSlave`, [25](#)
- `getId`
 - `BaseSlave`, [13](#)
 - `CO2Slave`, [20](#)

- RGBSlave, 25
- getSlave
 - SlaveManager, 47
- getStatus
 - BaseSlave, 14
 - CO2Slave, 20
 - RGBSlave, 25
- green_state
 - packets.h, 58
 - sensor_packet_rgb_light, 43
- header
 - packets.h, 58
 - sensor_packet, 36
- HEARTBEAT
 - packets.h, 57
- heartbeat
 - sensor_data, 28
 - sensor_packet::sensor_data, 30
- HUMIDITY
 - packets.h, 58
- humidity
 - sensor_data, 28
 - sensor_packet::sensor_data, 30
- i2c_address
 - RGBSlave, 26
- i2c_fd
 - SlaveManager, 47
- id
 - CO2Slave, 21
 - RGBSlave, 26
- include/BaseSlave.h, 49, 50
- include/BusServer.h, 50, 51
- include/CO2Slave.h, 52
- include/math.h, 53, 54
- include/packets.h, 55, 60
- include/RGBSlave.h, 61, 63
- include/SlaveManager.h, 64
- initialize
 - SlaveManager, 47
- initialized
 - SlaveManager, 48
- ledControl
 - SlaveManager, 47
- length
 - packets.h, 59
 - sensor_header, 31
- LIGHT
 - packets.h, 58
- light
 - sensor_data, 28
 - sensor_packet::sensor_data, 30
- listen
 - BusServer, 16
- listening_address
 - BusServer, 17
- listening_fd
 - BusServer, 17
- main
 - main.cpp, 70
- main.cpp
 - main, 70
- MASTER_ADDRESS
 - SlaveManager.cpp, 74
- math.cpp
 - add, 71
 - subtract, 72
- math.h
 - add, 54
 - subtract, 54
- metadata
 - packets.h, 59
 - sensor_heartbeat, 33
 - 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
- MOTION
 - packets.h, 58
- NOOP
 - packets.h, 58
- packets.h
 - __attribute__, 58
 - blue_state, 58
 - BUTTON, 58
 - CO2, 58
 - DASHBOARD_GET, 57
 - DASHBOARD_POST, 57
 - DASHBOARD_RESPONSE, 57
 - DATA, 57
 - data, 58
 - green_state, 58
 - header, 58
 - HEARTBEAT, 57
 - HUMIDITY, 58
 - length, 59
 - LIGHT, 58
 - metadata, 59
 - MOTION, 58
 - NOOP, 58
 - PacketType, 57
 - PRESSURE, 58
 - ptype, 59
 - red_state, 59
 - RGB_LIGHT, 58
 - sensor_id, 59
 - sensor_type, 59
 - SensorType, 57
 - target_state, 60
 - TEMPERATURE, 58
 - value, 60

- PacketType
 - packets.h, 57
- power_state
 - RGBSlave, 27
- PRESSURE
 - packets.h, 58
- ptype
 - packets.h, 59
 - sensor_header, 31
- R
 - RGBData, 22
 - RGBSlave.h, 63
- README.md, 65
- red_state
 - packets.h, 59
 - sensor_packet_rgb_light, 43
- RGB_LIGHT
 - packets.h, 58
- rgb_light
 - sensor_data, 28
 - sensor_packet::sensor_data, 30
- RGB_SLAVE_ADDRESS
 - BusServer.cpp, 66
- RGBData, 22
 - B, 22
 - G, 22
 - R, 22
- RGBSlave, 23
 - color_state, 26
 - fd, 26
 - getData, 25
 - getId, 25
 - getStatus, 25
 - i2c_address, 26
 - id, 26
 - power_state, 27
 - RGBSlave, 25
 - setData, 25
 - start, 26
 - stop, 26
- RGBSlave.h
 - __attribute__, 63
 - B, 63
 - G, 63
 - R, 63
- send
 - BusServer, 16
- sensor_data, 27
 - co2, 28
 - generic, 28
 - heartbeat, 28
 - humidity, 28
 - light, 28
 - rgb_light, 28
 - temperature, 28
- sensor_header, 31
 - length, 31
 - ptype, 31
- sensor_heartbeat, 32
 - metadata, 33
- sensor_id
 - packets.h, 59
 - sensor_metadata, 34
- sensor_metadata, 33
 - sensor_id, 34
 - sensor_type, 34
- sensor_packet, 34
 - data, 36
 - header, 36
- sensor_packet::sensor_data, 29
 - co2, 30
 - generic, 30
 - heartbeat, 30
 - humidity, 30
 - light, 30
 - rgb_light, 30
 - temperature, 30
- sensor_packet_co2, 36
 - metadata, 37
 - value, 37
- sensor_packet_generic, 38
 - metadata, 39
- sensor_packet_humidity, 39
 - metadata, 40
 - value, 40
- sensor_packet_light, 40
 - metadata, 41
 - target_state, 41
- sensor_packet_rgb_light, 42
 - blue_state, 43
 - green_state, 43
 - metadata, 43
 - red_state, 43
- sensor_packet_temperature, 44
 - metadata, 45
 - value, 45
- sensor_type
 - packets.h, 59
 - sensor_metadata, 34
- SensorType
 - packets.h, 57
- setData
 - BaseSlave, 14
 - CO2Slave, 20
 - RGBSlave, 25
- setup
 - BusServer, 16
- slave_manager
 - BusServer, 17
- SlaveManager, 45
 - address, 47
 - createSlave, 46
 - deleteSlave, 46
 - getSlave, 47
 - i2c_fd, 47

- initialize, [47](#)
 - initialized, [48](#)
 - ledControl, [47](#)
 - SlaveManager, [46](#)
 - slaves, [48](#)
- SlaveManager.cpp
 - MASTER_ADDRESS, [74](#)
- slaves
 - SlaveManager, [48](#)
- src/BaseSlave.cpp, [65](#), [66](#)
- src/BusServer.cpp, [66](#), [67](#)
- src/CO2Slave.cpp, [69](#)
- src/main.cpp, [69](#), [70](#)
- src/math.cpp, [71](#), [72](#)
- src/RGBSlave.cpp, [73](#)
- src/SlaveManager.cpp, [74](#), [75](#)
- start
 - BaseSlave, [14](#)
 - BusServer, [17](#)
 - CO2Slave, [20](#)
 - RGBSlave, [26](#)
- stop
 - BaseSlave, [14](#)
 - CO2Slave, [21](#)
 - RGBSlave, [26](#)
- subtract
 - math.cpp, [72](#)
 - math.h, [54](#)
- target_state
 - packets.h, [60](#)
 - sensor_packet_light, [41](#)
- TEMPERATURE
 - packets.h, [58](#)
- temperature
 - sensor_data, [28](#)
 - sensor_packet::sensor_data, [30](#)
- TEST
 - test_math.cpp, [77](#)
- Test List, [3](#)
- test_math.cpp
 - TEST, [77](#)
- tests/test_math.cpp, [75](#), [77](#)
- value
 - packets.h, [60](#)
 - sensor_packet_co2, [37](#)
 - sensor_packet_humidity, [40](#)
 - sensor_packet_temperature, [45](#)
- wemos_bridge_connected
 - BusServer, [18](#)