Dezibot 4

Generated by Doxygen 1.8.17

Chapter 1

Dezibot4 Lib

1.1 Links to external documentation

- PDF-Doku Code
- PDF-Doku Device

1.2 Link to Software

- Library

1.3 Introduction

The project focuses on the software for the Dezibot4 robot and its use in the classroom.
It includes libraries for use by students as well as guides for teachers. The hardware of the robot is not part of the project.
The libraries and example programs are available under the GPL and are accessible as a repository.
It is ment to serve as an Arduino-Library.
Therefore the rules for arduinolibrary develop apply:

- Styleguide
- Libraryspecification
- Submission-requirements

In the following the most important points and custom conventions are introduced.

1.4 Code Conventions

1.4.1 Don't pass reference

To allow easy usability for users not familier with C++, prevent passing around references. It is better to use accessmethods

1.4.2 Naming

- methods are named in lowerCamelCase
- classes are named in UpperCamelCase
- folders containing components are named in lowerCamelCase
- methods are named in lowerCamelCase
- constants are named in ALL_CAPS_SNAKE_CASE

1.4.3 Bytestream

Every class that implements Byte-Based Communication needs to implement the Arduino Streaminterface

1.4.4 Components

Every component has a single .h file and one or more .cpp files.
Every component is placed in a seperate folder under src/ that is named equvivalent to the class. The minimal structure of any .h file is

```
 {c++}
#ifndef ClassName_h
#define ClassName_h
class ClassName{
};
#endif //ClassName_h
```

1.5 Design Paradigm

During desgin, the Dezibot isn't describe using it's part but instead it's functionality. Under the top-level Dezibotclass, there is a class for every functionality of the robot. Each of that classes consists of two parts.

1.5.1 Part Instances

Each component contains instances of every Robotpart that is used in that component. For example the Motion component contains two motorinstances, one for motorEast and one for motorWest. Using these instances, it is possible to access more specific methods that interacts directly with the component (configure it, setSpeed,...)

1.5.2 Abstractions

The components constains abstractions that combines multiple partMethods to ease the usability. For example for the motioncomponent provides an abstraction for the forwardmovement, that involves two motors and even another component (MotionDetection)

1.6 Contributing

When contributing to the project please follow the rules below. At first, follow all rules from this readme. Further rules apply to the usage of git

1.6.1 Branching

Whenever working on the project, create a new branch from the current state of Develop. Branches should be named as `prefix/#issueid-shortdescription` where prefix is from {feature,fix,refactor}.
When a branch is ready to be used in production, create a mergerequest.

1.6.2 Mergerequests

The target of each Mergerequest must be the Develop-Branch. Before the merge, each request must be approved by at least one person with Owner role.
The approve process should consider especially the documentation, naming, implementation. When the merge is approved and no more commits are added, the last commit must increment the versionnumber in the library.↩ properties file, following the rules of `Semantic Versioning`

1.6.3 Commitmessages

Commitmessages must follow the `gitchangelog` pattern.

1.6.4 Language

The language of the project is American English. That includes in particular but not exclusively:

- Sourcecode
- Commit Messages
- Documentation

A german documentation will be provided but does not replace the english documentation.

1.6.5 Documentation

Documentation of the Software and Hardware can be found at `https://docs.dezibot.de/`

1.6.5.1 .h Files

```
 {C++}
/**
 * @file Dezibot.h
 * @author your name (you@domain.com)
 * @brief
 * @date 2023-11-19
 *
 * @copyright Copyright (c) 2023
 *
 */
```

In the library, the `.h` files should be included using a relative path. For instance, in `src/Dezibot.h`, to include `src/motion/Motion.h`, you should write `#include "motion/Motion.h"`.

1.6.5.2 Methods

```
 {C++}
/**
 * @brief
 * @param
 * ...
 * @return
 *
 */
```

1.6.5.3 Arduino Settings

- Board: "ESP32-S3-USB-OTG"
- Upload Mode: "UART0 / Hardware CDC"
- USB Mode: "Hardware CDC and JTAG"
- Programmer: "Esptool"

Using `arduino-cli` to compile and upload: `arduino-cli upload /Users/jo/Documents/↩ Arduino/theSketch -p /dev/cu.usbmodem101 -b esp32:esp32:nora_w10 arduino-cli compile /Users/jo/Documents/Arduino/theSketch -p /dev/cu.usbmodem101 -b esp32:esp32:nora_w10`

1.6.5.3.1 Including Library

Arduino IDE -> Sketch -> Include Library -> add .ZIP Library -> this library

If there is any other error like 'Painless_Mesh' not found, you have to include this library also.

Arduino IDE -> Sketch -> Manage Library -> Search for missing Library

1.6.5.4 Start from Scratch

It is important, before using any functions of Dezibot, to call `dezibot.begin()` once in the setup function.

In the examples folder, a sketch `start` is provided, that handles the initialization.

1.7 Third-Party Licenses

This project uses the following third-party libraries:

- `veml6040` (version 0.3.2) by `@thewknd` et al.
  - Vishay VEML6040 RGBW color sensor library for Arduino
  - Licensed under the MIT license
  - For more information, see the `library's repository`

Chapter 2

Dezibot4 library

- PDF-Doku Code
- PDF-Doku Device

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 averageMeasurement Struct Reference

`#include <LightDetection.h>`

Public Attributes

- photoTransistors sensor
- uint32_t measurementAmount
- uint32_t timeBetween
- uint16_t result
- bool done

5.1.1 Detailed Description

Definition at line 28 of file LightDetection.h.

5.1.2 Member Data Documentation

5.1.2.1 done

`bool averageMeasurement::done`

Definition at line 33 of file LightDetection.h.

5.1.2.2 measurementAmount

```
uint32_t averageMeasurement::measurementAmount
```

Definition at line 30 of file LightDetection.h.

5.1.2.3 result

```
uint16_t averageMeasurement::result
```

Definition at line 32 of file LightDetection.h.

5.1.2.4 sensor

```
photoTransistors averageMeasurement::sensor
```

Definition at line 29 of file LightDetection.h.

5.1.2.5 timeBetween

```
uint32_t averageMeasurement::timeBetween
```

Definition at line 31 of file LightDetection.h.

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

- src/lightDetection/LightDetection.h

5.2 ColorDetection Class Reference

```
#include <ColorDetection.h>
```

Public Member Functions

- void beginAutoMode ()
  Start RBGW sensor with default configuration.
- void configure (VEML_CONFIG config)
  Begin RGBW sensor with passed configuration values.
- uint16_t getColorValue (color color)
  Get color value of RGBW sensor.
- float getAmbientLight ()
  Get the ambient light in lux.

Protected Attributes

- VEML6040 rgbwSensor

5.2.1 Detailed Description

Definition at line 55 of file ColorDetection.h.

5.2.2 Member Function Documentation

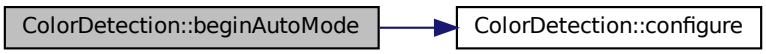
5.2.2.1 beginAutoMode()

```
void ColorDetection::beginAutoMode ( )
```

Start RBGW sensor with default configuration.

Definition at line 3 of file ColorDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.2.2.2 configure()

```
void ColorDetection::configure (
            VEML_CONFIG config )
```

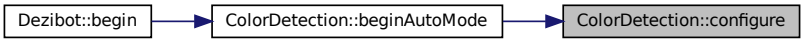
Begin RGBW sensor with passed configuration values.

Parameters

config	configuration for VEML6040 sensor

Definition at line 11 of file ColorDetection.cpp.

Here is the caller graph for this function:



5.2.2.3 getAmbientLight()

```
float ColorDetection::getAmbientLight ( )
```

Get the ambient light in lux.

Returns

float ambient light in lux.

Definition at line 59 of file ColorDetection.cpp.

5.2.2.4 getColorValue()

```
uint16_t ColorDetection::getColorValue (
            color color )
```

Get color value of RGBW sensor.

Parameters

color	RGBW color which to get

Returns

uint16_t color value

Definition at line 43 of file ColorDetection.cpp.

5.2.3 Member Data Documentation

5.2.3.1 rgbwSensor

```
VEML6040 ColorDetection::rgbwSensor  [protected]
```

Definition at line 86 of file ColorDetection.h.

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

- src/colorDetection/ColorDetection.h
- src/colorDetection/ColorDetection.cpp

5.3 Communication Class Reference

```
#include <Communication.h>
```

Public Member Functions

- void setGroupNumber (uint32_t number)
- void sendMessage (String msg)
- void onReceive (void(*callbackFunc)(String &msg))

Static Public Member Functions

- static void begin (void)

  initialize the Mesh Compnent, must be called before the other methods are used.

5.3.1 Detailed Description

Definition at line 13 of file Communication.h.

5.3.2 Member Function Documentation

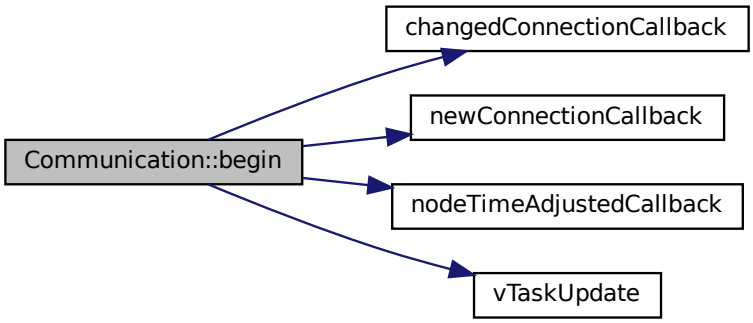
5.3.2.1 begin()

```
void Communication::begin (
            void  )  [static]
```

initialize the Mesh Compnent, must be called before the other methods are used.

Definition at line 69 of file Communication.cpp.

Here is the call graph for this function:



5.3.2.2 onReceive()

```
void Communication::onReceive (
            void(*)(String &msg) callbackFunc )
```

Definition at line 64 of file Communication.cpp.

5.3.2.3 sendMessage()

```
void Communication::sendMessage (
            String msg )
```

Definition at line 10 of file Communication.cpp.

5.3.2.4 setGroupNumber()

```
void Communication::setGroupNumber (
            uint32_t number )
```

Definition at line 59 of file Communication.cpp.

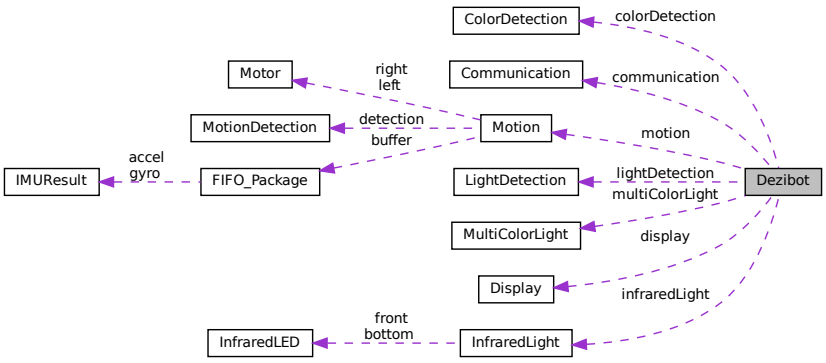
The documentation for this class was generated from the following files:

- src/communication/Communication.h
- src/communication/Communication.cpp

5.4 Dezibot Class Reference

```
#include <Dezibot.h>
```

Collaboration diagram for Dezibot:



Public Member Functions

- Dezibot ()
- void begin (void)

Public Attributes

- Motion motion
- LightDetection lightDetection
- ColorDetection colorDetection
- MultiColorLight multiColorLight
- InfraredLight infraredLight
- Communication communication
- Display display

5.4.1 Detailed Description

Definition at line 24 of file Dezibot.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Dezibot()

```
Dezibot::Dezibot ( )
```

Definition at line 9 of file Dezibot.cpp.

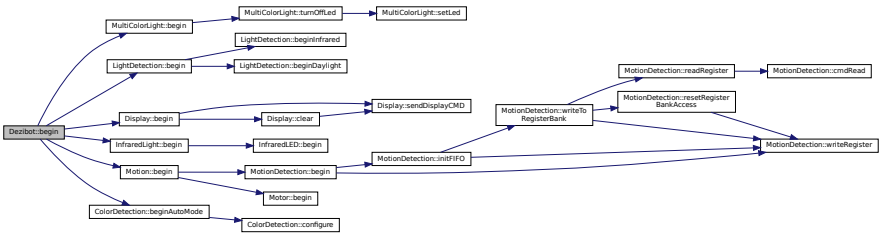
5.4.3 Member Function Documentation

5.4.3.1 begin()

```
void Dezibot::begin (
            void  )
```

Definition at line 11 of file Dezibot.cpp.

Here is the call graph for this function:



5.4.4 Member Data Documentation

5.4.4.1 colorDetection

```
ColorDetection Dezibot::colorDetection
```

Definition at line 31 of file Dezibot.h.

5.4.4.2 communication

```
Communication Dezibot::communication
```

Definition at line 34 of file Dezibot.h.

5.4.4.3 display

```
Display Dezibot::display
```

Definition at line 35 of file Dezibot.h.

5.4.4.4 infraredLight

```
InfraredLight Dezibot::infraredLight
```

Definition at line 33 of file Dezibot.h.

5.4.4.5 lightDetection

```
LightDetection Dezibot::lightDetection
```

Definition at line 30 of file Dezibot.h.

5.4.4.6 motion

```
Motion Dezibot::motion
```

Definition at line 29 of file Dezibot.h.

5.4.4.7 multiColorLight

```
MultiColorLight Dezibot::multiColorLight
```

Definition at line 32 of file Dezibot.h.

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

- src/Dezibot.h
- src/Dezibot.cpp

5.5 Display Class Reference

`#include <Display.h>`

Public Member Functions

- void begin (void)

  initializes the display datastructures and sents the required cmds to start the display. Should only be called once.
- void clear (void)

  delets all content from the display, resets the linecounter, new print will start at the top left. Orientationflip is not resetted
- void print (char ∗value)

  prints the passed string right behind the current displaycontent the sequence "\n" can be used to make a linebreak on the display
- void println (char ∗value)

  same as the print method, but after the string a line break is inserted
- void print (String value)

  prints the passed string right behind the current displaycontent the sequence "\n" can be used to make a linebreak on the display
- void println (String value)

  same as the print method, but after the string a line break is inserted
- void print (int value)

  prints the passed string right behind the current displaycontent the sequence "\n" can be used to make a linebreak on the display
- void println (int value)

  same as the print method, but after the string a line break is inserted
- char stringToCharArray (String value)

  string to char
- void flipOrientation (void)

  flips the horizontal orientation of all content on the display
- void invertColor (void)

  inverts the pixelcolors, so pixels on will be set to off and currently off pixels will be turned off. affects already printed content as well as future prints.

Protected Member Functions

- void sendDisplayCMD (uint8_t cmd)

  sends the passed cmd to the display, cmd_byte is added as prefix by the function
- void updateLine (uint charAmount)

  should be called whenever characters where printed to the display. Updates the data of the class to handle linebreaks correctly

Protected Attributes

- uint8_t charsOnCurrLine = 0
- uint8_t currLine = 0
- bool orientationFlipped = false
- bool colorInverted = false

5.5.1 Detailed Description

Definition at line 18 of file Display.h.

5.5.2 Member Function Documentation

5.5.2.1 begin()

```
void Display::begin (
             void  )
```

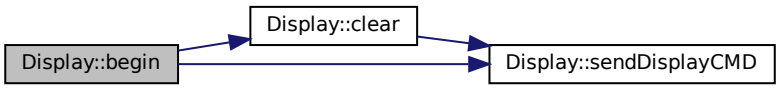
initializes the display datastructures and sents the required cmds to start the display. Should only be called once.

Warning

doesn't initalize the I^2C bus itself, therefore wire.begin(1,2) must be called elsewhere, before this method.

Definition at line 16 of file Display.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.5.2.2 clear()

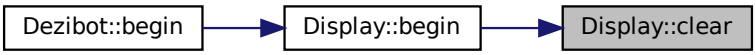
```
void Display::clear (
            void  )
```

delets all content from the display, resets the linecounter, new print will start at the top left. Orientationflip is not resetted

Definition at line 44 of file Display.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



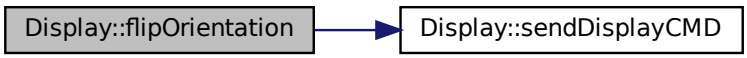
5.5.2.3 flipOrientation()

```
void Display::flipOrientation (
            void  )
```

flips the horizontal orientation of all content on the display

Definition at line 157 of file Display.cpp.

Here is the call graph for this function:



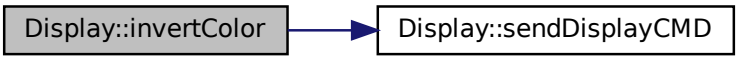
5.5.2.4 invertColor()

```
void Display::invertColor (
            void  )
```

inverts the pixelcolors, so pixels on will be set to off and currently off pixels will be turned off. affects already printed content as well as future prints.

Definition at line 168 of file Display.cpp.

Here is the call graph for this function:



5.5.2.5 print() [1/3]

```
void Display::print (
            char * value )
```

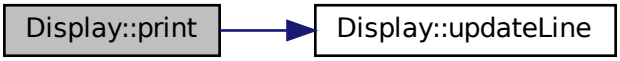
prints the passed string right behind the current displaycontent the sequence "\n" can be used to make a linebreak on the display

Parameters

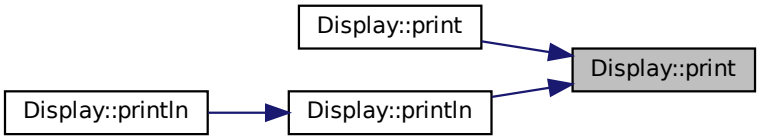
value	the string "xyz" that should be printed to the display

Definition at line 79 of file Display.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.5.2.6 print() [2/3]

```
void Display::print (
            int value )
```

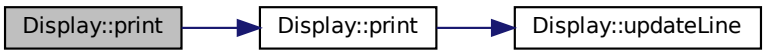
prints the passed string right behind the current displaycontent the sequence "\n" can be used to make a linebreak on the display

Parameters

value	the string "xyz" that should be printed to the display

Definition at line 140 of file Display.cpp.

Here is the call graph for this function:



5.5.2.7 print() [3/3]

```
void Display::print (
            String value )
```

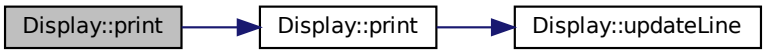
prints the passed string right behind the current displaycontent the sequence "\n" can be used to make a linebreak on the display

Parameters

value	the string "xyz" that should be printed to the display

Definition at line 124 of file Display.cpp.

Here is the call graph for this function:



5.5.2.8 println() [1/3]

```
void Display::println (
            char * value )
```

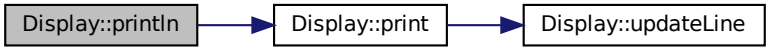
same as the print method, but after the string a line break is inserted

Parameters

value	the string that should be printed

Definition at line 152 of file Display.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.5.2.9 println() [2/3]

```
void Display::println (
            int value )
```

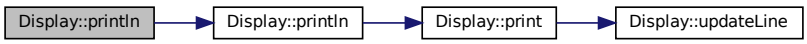
same as the print method, but after the string a line break is inserted

Parameters

value	the string that should be printed

Definition at line 146 of file Display.cpp.

Here is the call graph for this function:



5.5.2.10 println() [3/3]

```
void Display::println (
            String value )
```

same as the print method, but after the string a line break is inserted

Parameters

value	the string that should be printed

Definition at line 132 of file Display.cpp.

Here is the call graph for this function:

5.5.2.11 sendDisplayCMD()

```
void Display::sendDisplayCMD (
            uint8_t cmd )  [protected]
```

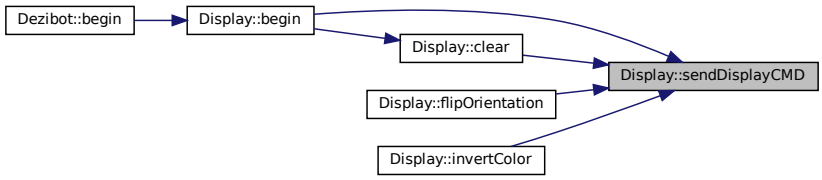
sends the passed cmd to the display, cmd_byte is added as prefix by the function

Parameters

cmd	the byte instruction that shold by sent

Definition at line 37 of file Display.cpp.

Here is the caller graph for this function:



5.5.2.12 stringToCharArray()

```
char Display::stringToCharArray (
            String value )
```

string to char

Parameters

value	the string that should be converted to char

Definition at line 115 of file Display.cpp.

5.5.2.13 updateLine()

```
void Display::updateLine (
            uint charAmount )  [protected]
```

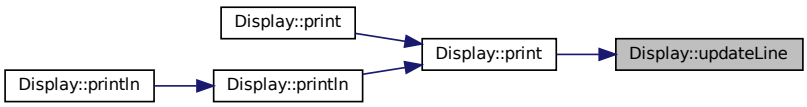
should be called whenever characters where printed to the display. Updates the data of the class to handle line-breaks correctly

Parameters

charAmount	How many characters where added to the screen

Definition at line 66 of file Display.cpp.

Here is the caller graph for this function:



5.5.3 Member Data Documentation

5.5.3.1 charsOnCurrLine

```
uint8_t Display::charsOnCurrLine = 0  [protected]
```

Definition at line 21 of file Display.h.

5.5.3.2 colorInverted

```
bool Display::colorInverted = false  [protected]
```

Definition at line 30 of file Display.h.

5.5.3.3 currLine

```
uint8_t Display::currLine = 0  [protected]
```

Definition at line 24 of file Display.h.

5.5.3.4 orientationFlipped

```
bool Display::orientationFlipped = false  [protected]
```

Definition at line 27 of file Display.h.

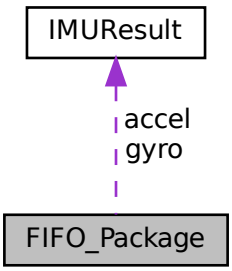
The documentation for this class was generated from the following files:

- src/display/Display.h
- src/display/Display.cpp

5.6 FIFO_Package Struct Reference

```
#include <MotionDetection.h>
```

Collaboration diagram for FIFO_Package:



Public Attributes

- int8_t header
- IMUResult gyro
- IMUResult accel
- int16_t temperature
- int16_t timestamp

5.6.1 Detailed Description

Definition at line 42 of file MotionDetection.h.

5.6.2 Member Data Documentation

5.6.2.1 accel

```
IMUResult FIFO_Package::accel
```

Definition at line 45 of file MotionDetection.h.

5.6.2.2 gyro

```
IMUResult FIFO_Package::gyro
```

Definition at line 44 of file MotionDetection.h.

5.6.2.3 header

```
int8_t FIFO_Package::header
```

Definition at line 43 of file MotionDetection.h.

5.6.2.4 temperature

```
int16_t FIFO_Package::temperature
```

Definition at line 46 of file MotionDetection.h.

5.6.2.5 timestamp

```
int16_t FIFO_Package::timestamp
```

Definition at line 47 of file MotionDetection.h.

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

- src/motionDetection/MotionDetection.h

5.7 IMUResult Struct Reference

```
#include <MotionDetection.h>
```

Public Attributes

- int16_t x
- int16_t y
- int16_t z

5.7.1 Detailed Description

Definition at line 16 of file MotionDetection.h.

5.7.2 Member Data Documentation

5.7.2.1 x

```
int16_t IMUResult::x
```

Definition at line 17 of file MotionDetection.h.

5.7.2.2 y

```
int16_t IMUResult::y
```

Definition at line 18 of file MotionDetection.h.

5.7.2.3 z

```
int16_t IMUResult::z
```

Definition at line 19 of file MotionDetection.h.

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

- src/motionDetection/MotionDetection.h

5.8 InfraredLED Class Reference

```
#include <InfraredLight.h>
```

Public Member Functions

- InfraredLED (uint8_t pin, ledc_timer_t timer, ledc_channel_t channel)
- void begin (void)
- void turnOn (void)

  enables selected LED
- void turnOff (void)

  disables selected LED
- void setState (bool state)

  changes state of selected LED depending on the state
- void sendFrequency (uint16_t frequency)

  starts flashing the IRLed with a specific frequency Won't stop automatically, must be stopped by calling any other IR-Method

Protected Attributes

- uint8_t ledPin
- ledc_timer_t timer
- ledc_channel_t channel
- ledc_timer_config_t pwmTimer
- ledc_channel_config_t pwmChannel

5.8.1 Detailed Description

Definition at line 18 of file InfraredLight.h.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 InfraredLED()

```
InfraredLED::InfraredLED (
            uint8_t pin,
            ledc_timer_t timer,
            ledc_channel_t channel )
```

Definition at line 5 of file InfraredLED.cpp.

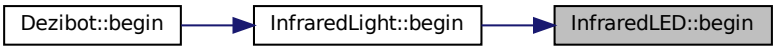
5.8.3 Member Function Documentation

5.8.3.1 begin()

```
void InfraredLED::begin (
            void  )
```

Definition at line 11 of file InfraredLED.cpp.

Here is the caller graph for this function:



5.8.3.2 sendFrequency()

```
void InfraredLED::sendFrequency (
            uint16_t frequency )
```

starts flashing the IRLed with a specific frequency Won't stop automatically, must be stopped by calling any other IR-Method

Parameters

frequency	

Definition at line 53 of file InfraredLED.cpp.

5.8.3.3 setState()

```
void InfraredLED::setState (
            bool state )
```

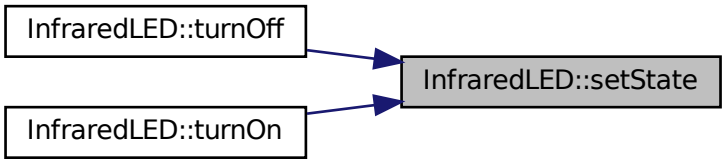
changes state of selected LED depending on the state

Parameters

led	which led will be affected
state	true if led should be turned on, else false

Definition at line 42 of file InfraredLED.cpp.

Here is the caller graph for this function:



5.8.3.4 turnOff()

```
void InfraredLED::turnOff (
            void  )
```

disables selected LED

Parameters

led	

Definition at line 38 of file InfraredLED.cpp.

Here is the call graph for this function:

5.8.3.5 turnOn()

```
void InfraredLED::turnOn (
            void  )
```

enables selected LED

Definition at line 34 of file InfraredLED.cpp.

Here is the call graph for this function:

5.8.4 Member Data Documentation

5.8.4.1 channel

```
ledc_channel_t InfraredLED::channel  [protected]
```

Definition at line 50 of file InfraredLight.h.

5.8.4.2 ledPin

```
uint8_t InfraredLED::ledPin  [protected]
```

Definition at line 48 of file InfraredLight.h.

5.8.4.3 pwmChannel

```
ledc_channel_config_t InfraredLED::pwmChannel  [protected]
```

Definition at line 52 of file InfraredLight.h.

5.8.4.4 pwmTimer

```
ledc_timer_config_t InfraredLED::pwmTimer  [protected]
```

Definition at line 51 of file InfraredLight.h.

5.8.4.5 timer

```
ledc_timer_t InfraredLED::timer  [protected]
```

Definition at line 49 of file InfraredLight.h.

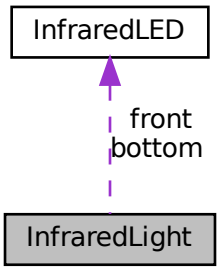
The documentation for this class was generated from the following files:

- src/infraredLight/InfraredLight.h
- src/infraredLight/InfraredLED.cpp

5.9 InfraredLight Class Reference

```
#include <InfraredLight.h>
```

Collaboration diagram for InfraredLight:



Public Member Functions

- void begin (void)

Public Attributes

- InfraredLED bottom = InfraredLED(IRBottomPin,LEDC_TIMER_0,LEDC_CHANNEL_0)
- InfraredLED front = InfraredLED(IRFrontPin,LEDC_TIMER_1,LEDC_CHANNEL_1)

Static Protected Attributes

- static const uint8_t IRFrontPin = 14
- static const uint8_t IRBottomPin = 13

5.9.1 Detailed Description

Definition at line 55 of file InfraredLight.h.

5.9.2 Member Function Documentation

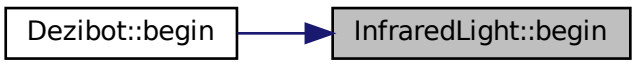
5.9.2.1 begin()

```
void InfraredLight::begin (
            void  )
```

Definition at line 3 of file InfraredLight.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



5.9.3 Member Data Documentation

5.9.3.1 bottom

```
InfraredLED InfraredLight::bottom = InfraredLED(IRBottomPin,LEDC_TIMER_0,LEDC_CHANNEL_0)
```

Definition at line 58 of file InfraredLight.h.

5.9.3.2 front

```
InfraredLED InfraredLight::front = InfraredLED(IRFrontPin,LEDC_TIMER_1,LEDC_CHANNEL_1)
```

Definition at line 59 of file InfraredLight.h.

5.9.3.3 IRBottomPin

```
const uint8_t InfraredLight::IRBottomPin = 13  [static], [protected]
```

Definition at line 64 of file InfraredLight.h.

5.9.3.4 IRFrontPin

```
const uint8_t InfraredLight::IRFrontPin = 14  [static], [protected]
```

Definition at line 63 of file InfraredLight.h.

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

- src/infraredLight/InfraredLight.h
- src/infraredLight/InfraredLight.cpp

5.10 LightDetection Class Reference

```
#include <LightDetection.h>
```

Static Public Member Functions

- static void begin (void)

  initialize the Lightdetection Compnent, must be called before the other methods are used.

- static uint16_t getValue (photoTransistors sensor)

  reads the Value of the specified sensor

- static photoTransistors getBrightest (ptType type)

  can be used to determine which sensor is exposed to the greatest amount of light Can distingish between IR and Daylight

- static uint32_t getAverageValue (photoTransistors sensor, uint32_t measurments, uint32_t timeBetween)

  Get the Average of multiple measurments of a single PT.

Static Protected Member Functions

- static void beginInfrared (void)
- static void beginDaylight (void)
- static uint16_t readIRPT (photoTransistors sensor)
- static uint16_t readDLPT (photoTransistors sensor)

Static Protected Attributes

- static const uint8_t IR_PT_FRONT_ADC = 3
- static const uint8_t IR_PT_LEFT_ADC = 4
- static const uint8_t IR_PT_RIGHT_ADC = 5
- static const uint8_t IR_PT_BACK_ADC = 6
- static const uint8_t DL_PT_FRONT_ADC = 7
- static const uint8_t DL_PT_BOTTOM_ADC = 8
- static const uint8_t DL_PT_ENABLE = 41
- static const uint8_t IR_PT_ENABLE = 40

5.10.1 Detailed Description

Definition at line 44 of file LightDetection.h.

5.10.2 Member Function Documentation

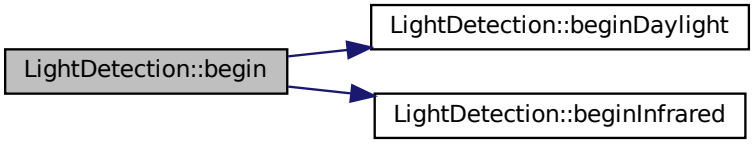
5.10.2.1 begin()

```
void LightDetection::begin (
            void  )  [static]
```

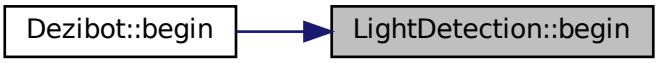
initialize the Lightdetection Compnent, must be called before the other methods are used.

Definition at line 4 of file LightDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

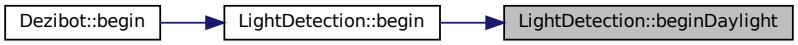


5.10.2.2 beginDaylight()

```
void LightDetection::beginDaylight (
            void  )  [static], [protected]
```

Definition at line 75 of file LightDetection.cpp.

Here is the caller graph for this function:

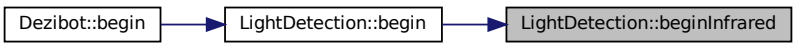


5.10.2.3 beginInfrared()

```
void LightDetection::beginInfrared (
            void  )  [static], [protected]
```

Definition at line 66 of file LightDetection.cpp.

Here is the caller graph for this function:



5.10.2.4 getAverageValue()

```
uint32_t LightDetection::getAverageValue (
            photoTransistors sensor,
            uint32_t measurments,
            uint32_t timeBetween )  [static]
```

Get the Average of multiple measurments of a single PT.

Parameters

sensor	Which Phototransistor should be read
measurments	how many measurements should be taken
timeBetween	which time should elapse between

Returns

the average of all taken meaurments

Definition at line 54 of file LightDetection.cpp.

Here is the call graph for this function:

5.10.2.5 getBrightest()

```
photoTransistors LightDetection::getBrightest (
            ptType type )  [static]
```

can be used to determine which sensor is exposed to the greatest amount of light Can distingish between IR and Daylight

Parameters

type	select which PTTransistors to compare

Returns

photoTransistors which sensor is exposed to the greatest amount of light, if all sensor read 0, the front sensor is returned

Definition at line 26 of file LightDetection.cpp.

5.10.2.6 getValue()

```
uint16_t LightDetection::getValue (
            photoTransistors sensor )  [static]
```

reads the Value of the specified sensor

Parameters

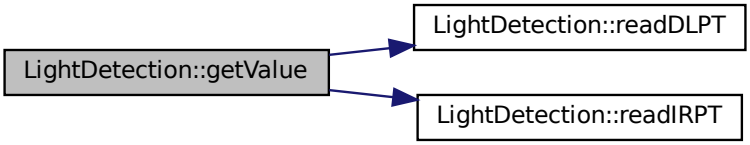
sensor	which sensor to read

Returns

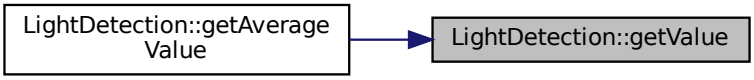
uint the reading of the sensor. between 0-4095

Definition at line 9 of file LightDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.7 readDLPT()

```
uint16_t LightDetection::readDLPT (
            photoTransistors sensor )  [static], [protected]
```

Definition at line 106 of file LightDetection.cpp.

Here is the caller graph for this function:

5.10.2.8 readIRPT()

```
uint16_t LightDetection::readIRPT (
            photoTransistors sensor )  [static], [protected]
```

Definition at line 82 of file LightDetection.cpp.

Here is the caller graph for this function:

5.10.3 Member Data Documentation

5.10.3.1 DL_PT_BOTTOM_ADC

```
const uint8_t LightDetection::DL_PT_BOTTOM_ADC = 8  [static], [protected]
```

Definition at line 85 of file LightDetection.h.

5.10.3.2 DL_PT_ENABLE

```
const uint8_t LightDetection::DL_PT_ENABLE = 41  [static], [protected]
```

Definition at line 87 of file LightDetection.h.

5.10.3.3 DL_PT_FRONT_ADC

```
const uint8_t LightDetection::DL_PT_FRONT_ADC = 7  [static], [protected]
```

Definition at line 84 of file LightDetection.h.

5.10.3.4 IR_PT_BACK_ADC

```
const uint8_t LightDetection::IR_PT_BACK_ADC = 6  [static], [protected]
```

Definition at line 82 of file LightDetection.h.

5.10.3.5 IR_PT_ENABLE

```
const uint8_t LightDetection::IR_PT_ENABLE = 40  [static], [protected]
```

Definition at line 88 of file LightDetection.h.

5.10.3.6 IR_PT_FRONT_ADC

```
const uint8_t LightDetection::IR_PT_FRONT_ADC = 3  [static], [protected]
```

Definition at line 79 of file LightDetection.h.

5.10.3.7 IR_PT_LEFT_ADC

```
const uint8_t LightDetection::IR_PT_LEFT_ADC = 4  [static], [protected]
```

Definition at line 80 of file LightDetection.h.

5.10.3.8 IR_PT_RIGHT_ADC

```
const uint8_t LightDetection::IR_PT_RIGHT_ADC = 5  [static], [protected]
```

Definition at line 81 of file LightDetection.h.

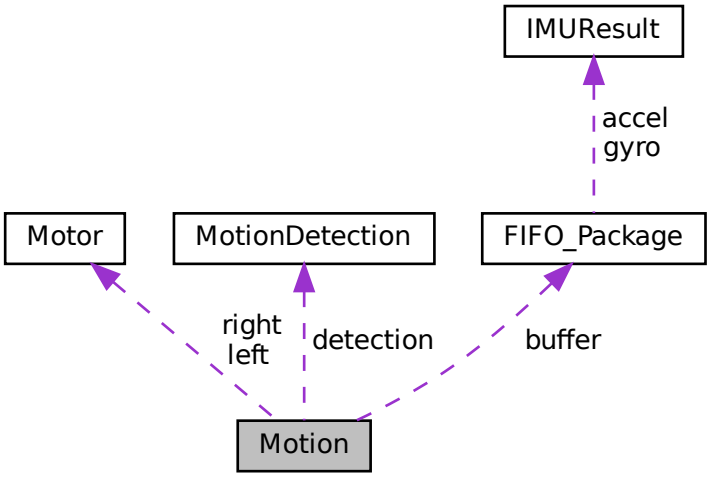
The documentation for this class was generated from the following files:

- src/lightDetection/LightDetection.h
- src/lightDetection/LightDetection.cpp

5.11 Motion Class Reference

`#include <Motion.h>`

Collaboration diagram for Motion:



Public Member Functions

- void begin (void)

  Initialize the movement component.

Static Public Member Functions

- static void move (uint32_t moveForMs=0, uint baseValue=DEFAULT_BASE_VALUE)

  Move forward for a certain amount of time. Call with moveForMs 0 will start movement, that must be stopped explicit by call to stop(). The function applys a basic algorithm to improve the straigthness of the movement. Lifting the robot from the desk may corrupt the results and is not recommended.
- static void rotateClockwise (uint32_t rotateForMs=0, uint baseValue=DEFAULT_BASE_VALUE)

  Rotate clockwise for a certain amount of time. Call with moveForMs 0 will start movement, that must be stopped explicit by call to stop().
- static void rotateAntiClockwise (uint32_t rotateForMs=0, uint baseValue=DEFAULT_BASE_VALUE)

  Rotate anticlockwise for a certain amount of time. Call with moveForMs 0 will start movement, that must be stopped explicit by call to stop().
- static void stop (void)

  stops any current movement, no matter if timebased or endless
- static void moveWithoutCorrection (uint32_t moveForMs=0, uint baseValue=DEFAULT_BASE_VALUE)

  Does the same as the move function, but this function does not apply any kind of algorithm to improve the result.

Static Public Attributes

- static Motor left = Motor(MOTOR_LEFT_PIN,TIMER,CHANNEL_LEFT)
- static Motor right = Motor(MOTOR_RIGHT_PIN,TIMER,CHANNEL_RIGHT)
- static MotionDetection detection

Static Protected Member Functions

- static void moveTask (void *args)
- static void leftMotorTask (void *args)
- static void rightMotorTask (void *args)

Static Protected Attributes

- static uint16_t RIGHT_MOTOR_DUTY = DEFAULT_BASE_VALUE
- static uint16_t LEFT_MOTOR_DUTY = DEFAULT_BASE_VALUE
- static const int MOTOR_RIGHT_PIN = 11
- static const int MOTOR_LEFT_PIN = 12
- static TaskHandle_t xMoveTaskHandle = NULL
- static TaskHandle_t xClockwiseTaskHandle = NULL
- static TaskHandle_t xAntiClockwiseTaskHandle = NULL
- static TickType_t xLastWakeTime
- static FIFO_Package * buffer = new FIFO_Package[64]
- static int correctionThreshold = 150

5.11.1 Detailed Description

Definition at line 59 of file Motion.h.

5.11.2 Member Function Documentation

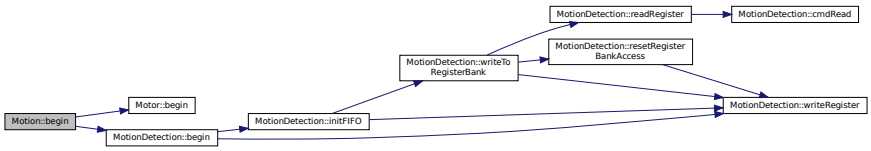
5.11.2.1 begin()

```
void Motion::begin (
            void  )
```

Initialize the movement component.

Definition at line 18 of file Motion.cpp.

Here is the call graph for this function:



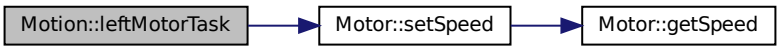
Here is the caller graph for this function:

5.11.2.2 leftMotorTask()

```
void Motion::leftMotorTask (
            void * args )  [static], [protected]
```

Definition at line 108 of file Motion.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.11.2.3 move()

```
void Motion::move (
            uint32_t moveForMs = 0,
            uint baseValue = DEFAULT_BASE_VALUE )  [static]
```

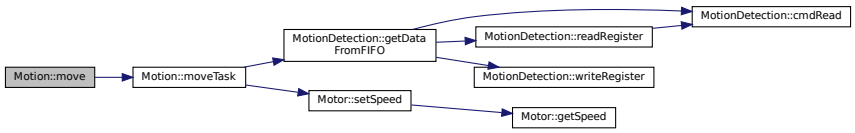
Move forward for a certain amount of time. Call with moveForMs 0 will start movement, that must be stopped explicit by call to stop(). The function applys a basic algorithm to improve the straigthness of the movement. Lifting the robot from the desk may corrupt the results and is not recommended.

Parameters

moveForMs	Representing the duration of forward moving in milliseconds.
baseValue	The value that is used to start with the calibrated movement. Defaults to 3900. If the Dezibot is not moving forward at all increasing the value may help. If the robot is just jumping up and down but not forward, try a lower value.

Definition at line 87 of file Motion.cpp.

Here is the call graph for this function:

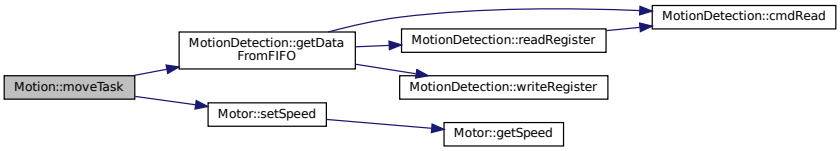


5.11.2.4 moveTask()

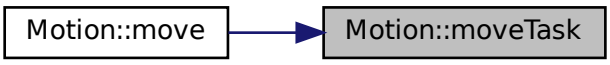
```
void Motion::moveTask (
            void * args )  [static], [protected]
```

Definition at line 31 of file Motion.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.2.5 moveWithoutCorrection()

```
static void Motion::moveWithoutCorrection (
            uint32_t moveForMs = 0,
            uint baseValue = DEFAULT_BASE_VALUE )  [static]
```

Does the same as the move function, but this function does not apply any kind of algorithm to improve the result.

Parameters

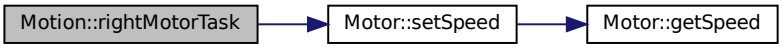
moveForMs	how many ms should the robot move, or 0 to let the robot move until another move command is mentioned, default is 0
baseValue	the duty value that is used for the movement, default is 0

5.11.2.6 rightMotorTask()

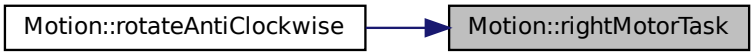
```
void Motion::rightMotorTask (
            void * args )  [static], [protected]
```

Definition at line 148 of file Motion.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.2.7 rotateAntiClockwise()

```
void Motion::rotateAntiClockwise (
            uint32_t rotateForMs = 0,
            uint baseValue = DEFAULT_BASE_VALUE )  [static]
```

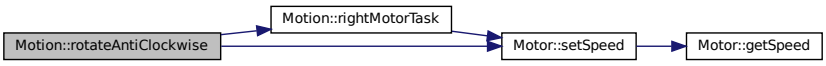
Rotate anticlockwise for a certain amount of time. Call with moveForMs 0 will start movement, that must be stopped explicit by call to stop().

Parameters

rotateForMs	Representing the duration of rotating anticlockwise in milliseconds or 0 to let the robot turn until another movecommand is issued. Default is 0.
baseValue	The value that is used to start with the calibrated movement (not released yet, currently just the used value).

Definition at line 173 of file Motion.cpp.

Here is the call graph for this function:



5.11.2.8 rotateClockwise()

```
void Motion::rotateClockwise (
            uint32_t rotateForMs = 0,
            uint baseValue = DEFAULT_BASE_VALUE )  [static]
```

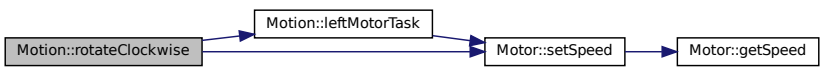
Rotate clockwise for a certain amount of time. Call with moveForMs 0 will start movement, that must be stopped explicit by call to stop().

Parameters

rotateForMs	Representing the duration of rotating clockwise in milliseconds, or 0 to rotate until another movecmd is issued. Default is 0
baseValue	The value that is used to start with the calibrated movement (not released yet, currently just the used value)

Definition at line 134 of file Motion.cpp.

Here is the call graph for this function:



5.11.2.9 stop()

```
void Motion::stop (
            void  )  [static]
```

stops any current movement, no matter if timebased or endless

Definition at line 187 of file Motion.cpp.

Here is the call graph for this function:

5.11.3 Member Data Documentation

5.11.3.1 buffer

```
FIFO_Package* Motion::buffer = new FIFO_Package[64]  [inline], [static], [protected]
```

Definition at line 73 of file Motion.h.

5.11.3.2 correctionThreshold

```
int Motion::correctionThreshold = 150  [inline], [static], [protected]
```

Definition at line 74 of file Motion.h.

5.11.3.3 detection

```
MotionDetection Motion::detection  [inline], [static]
```

Definition at line 82 of file Motion.h.

5.11.3.4 left

```
Motor Motion::left = Motor(MOTOR_LEFT_PIN,TIMER,CHANNEL_LEFT)  [inline], [static]
```

Definition at line 78 of file Motion.h.

5.11.3.5 LEFT_MOTOR_DUTY

```
uint16_t Motion::LEFT_MOTOR_DUTY = DEFAULT_BASE_VALUE  [inline], [static], [protected]
```

Definition at line 62 of file Motion.h.

5.11.3.6 MOTOR_LEFT_PIN

```
const int Motion::MOTOR_LEFT_PIN = 12  [static], [protected]
```

Definition at line 64 of file Motion.h.

5.11.3.7 MOTOR_RIGHT_PIN

```
const int Motion::MOTOR_RIGHT_PIN = 11  [static], [protected]
```

Definition at line 63 of file Motion.h.

5.11.3.8 right

```
Motor Motion::right = Motor(MOTOR_RIGHT_PIN,TIMER,CHANNEL_RIGHT)  [inline], [static]
```

Definition at line 79 of file Motion.h.

5.11.3.9 RIGHT_MOTOR_DUTY

```
uint16_t Motion::RIGHT_MOTOR_DUTY = DEFAULT_BASE_VALUE  [inline], [static], [protected]
```

Definition at line 61 of file Motion.h.

5.11.3.10 xAntiClockwiseTaskHandle

```
TaskHandle_t Motion::xAntiClockwiseTaskHandle = NULL  [inline], [static], [protected]
```

Definition at line 70 of file Motion.h.

5.11.3.11 xClockwiseTaskHandle

```
TaskHandle_t Motion::xClockwiseTaskHandle = NULL  [inline], [static], [protected]
```

Definition at line 69 of file Motion.h.

5.11.3.12 xLastWakeTime

```
TickType_t Motion::xLastWakeTime  [inline], [static], [protected]
```

Definition at line 71 of file Motion.h.

5.11.3.13 xMoveTaskHandle

```
TaskHandle_t Motion::xMoveTaskHandle = NULL  [inline], [static], [protected]
```

Definition at line 68 of file Motion.h.

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

- src/motion/Motion.h
- src/motion/Motion.cpp

5.12 MotionDetection Class Reference

```
#include <MotionDetection.h>
```

Public Member Functions

- MotionDetection ()
- void begin (void)

  initialized the IMU Component. Wakes the IMU from Standby Set configuration
- void end (void)

  stops the component Sets the IMU to Low-Power-Mode
- IMUResult getAcceleration (void)

  Triggers a new Reading of the accelerationvalues and reads them from the IMU.
- IMUResult getRotation (void)

  Triggers a new reading of the gyroscope and reads the values from the imu.
- float getTemperature (void)

  Reads the current On Chip temperature of the IMU.
- int8_t getWhoAmI (void)

  Returns the value of reading the whoAmI register When IMU working correctly, value should be 0x67.
- bool isShaken (uint32_t threshold=defaultShakeThreshold, uint8_t axis=xAxis|yAxis|zAxis)

  Detects if at the time of calling is shaken. Therefore the sum over all accelerationvalues is calculated and checked against threshold. If sum > threshold a shake is detected, else not.
- Orientation getTilt ()

  calculates how the robot is tilted. It is set, that when the robot is placed normally on a flat table, the result will be (0,0) Tilting the robot, so that the front leg is deeper than the other to results in an increasing degrees, tilting the front leg up will increase negativ degrees Tilting the robot to the right will increase the degrees until 180°(upside down), tilting it left will result in increasing negativ degrees (-1,-2,...,-180). On the top there is a jump of the values from 180->-180 and vice versa.
- Direction getTiltDirection (uint tolerance=10)

  Checks in which direction (Front, Left, Right, Back) the robot is tilted.
- void calibrateZAxis (uint gforceValue)
- uint getDataFromFIFO (FIFO_Package *buffer)

  will read all availible packages from fifo, after 40ms Fifo is full

Protected Types

- enum registerBank { MREG1, MREG2, MREG3 }

Protected Member Functions

- uint8_t readFromRegisterBank (registerBank bank, uint8_t reg)
- void writeToRegisterBank (registerBank bank, uint8_t reg, uint8_t value)
- void resetRegisterBankAccess ()
- uint16_t cmdRead (uint8_t regHigh, uint8_t regLow)
- uint16_t cmdWrite (uint8_t regHigh, uint8_t regLow)
- uint8_t cmdRead (uint8_t reg)
- uint8_t cmdWrite (uint8_t reg)
- uint8_t readRegister (uint8_t reg)
- int16_t readDoubleRegister (uint8_t lowerReg)
- void writeRegister (uint8_t reg, uint8_t value)
- void initFIFO ()

Protected Attributes

- const uint bufferLength = 64∗16
- int8_t ∗ buf = new int8_t[bufferLength]
- SPIClass ∗ handler = NULL
- uint gForceCalib = 4050

Static Protected Attributes

- static const uint frequency = 24000000
- static const uint16_t defaultShakeThreshold = 500

5.12.1 Detailed Description

Definition at line 51 of file MotionDetection.h.

5.12.2 Member Enumeration Documentation

5.12.2.1 registerBank

```
enum MotionDetection::registerBank  [protected]
```

Enumerator

MREG1	
MREG2	
MREG3	

Definition at line 53 of file MotionDetection.h.

5.12.3 Constructor & Destructor Documentation

5.12.3.1 MotionDetection()

```
MotionDetection::MotionDetection ( )
```

Definition at line 4 of file MotionDetection.cpp.

5.12.4 Member Function Documentation

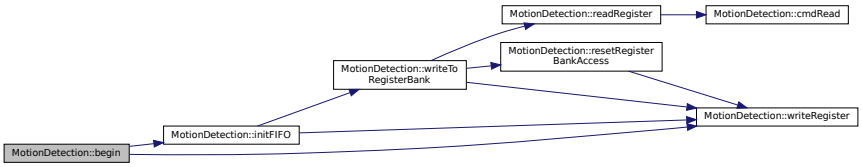
5.12.4.1 begin()

```
void MotionDetection::begin (
            void  )
```

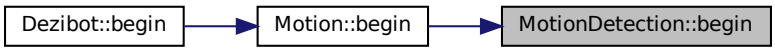
initialized the IMU Component. Wakes the IMU from Standby Set configuration

Definition at line 8 of file MotionDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.4.2 calibrateZAxis()

```
void MotionDetection::calibrateZAxis (
            uint gforceValue )
```

can be used to set a custom value for the gforceReading of the zaxis, which will improve the getTiltFunction.

Attention

this method is not persisten, so the value is not stored when the programm is restarted / the robot is powerd off

Parameters

gforceValue	the value the IMU returns for the gravitationforce -> to get this value, place the robot on a leveled surface and read the value getAcceleration().z

Definition at line 74 of file MotionDetection.cpp.

5.12.4.3 cmdRead() [1/2]

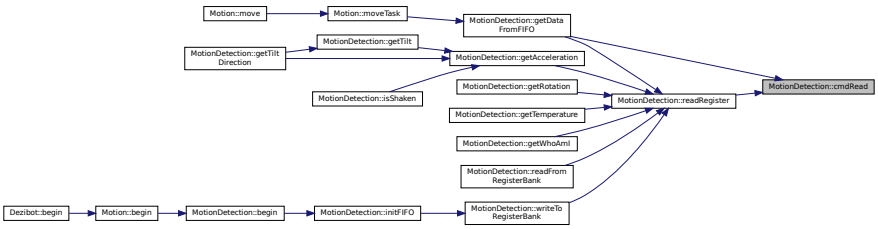
```
uint8_t MotionDetection::cmdRead (
            uint8_t reg )  [protected]
```

Definition at line 155 of file MotionDetection.cpp.

5.12.4.4 cmdRead() [2/2]

```
uint16_t MotionDetection::cmdRead (
            uint8_t regHigh,
            uint8_t regLow )  [protected]
```

Here is the caller graph for this function:



5.12.4.5 cmdWrite() [1/2]

```
uint8_t MotionDetection::cmdWrite (
            uint8_t reg )  [protected]
```

Definition at line 158 of file MotionDetection.cpp.

5.12.4.6 cmdWrite() [2/2]

```
uint16_t MotionDetection::cmdWrite (
            uint8_t regHigh,
            uint8_t regLow )  [protected]
```

5.12.4.7 end()

```
void MotionDetection::end (
            void  )
```

stops the component Sets the IMU to Low-Power-Mode

Definition at line 25 of file MotionDetection.cpp.

Here is the call graph for this function:

5.12.4.8 getAcceleration()

```
IMUResult MotionDetection::getAcceleration (
            void  )
```

Triggers a new Reading of the accelerationvalues and reads them from the IMU.

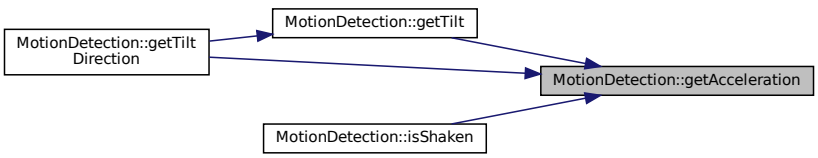
Returns

IMUResult that contains the new read values

Definition at line 28 of file MotionDetection.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



5.12.4.9 getDataFromFIFO()

```
uint MotionDetection::getDataFromFIFO (
            FIFO_Package * buffer )
```

will read all availible packages from fifo, after 40ms Fifo is full

Parameters

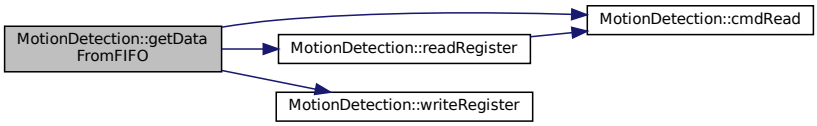
buffer	pointer to FIFO_Package Struct that at least must have size 64 (this is the max package count with APEX Enabled)

Returns

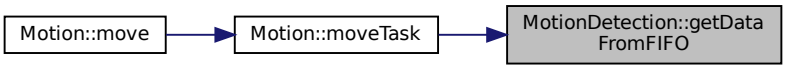
the amount of acutally fetched packages

Definition at line 245 of file MotionDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.4.10 getRotation()

```
IMUResult MotionDetection::getRotation (
            void  )
```

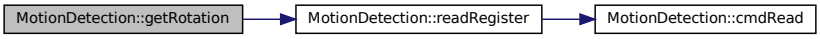
Triggers a new reading of the gyroscope and reads the values from the imu.

Returns

IMUResult

Definition at line 35 of file MotionDetection.cpp.

Here is the call graph for this function:



5.12.4.11 getTemperature()

```
float MotionDetection::getTemperature (
            void  )
```

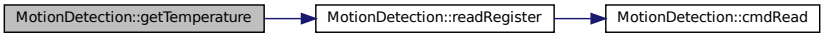
Reads the current On Chip temperature of the IMU.

Returns

normalized temperature in degree Centigrade

Definition at line 45 of file MotionDetection.cpp.

Here is the call graph for this function:



5.12.4.12 getTilt()

```
Orientation MotionDetection::getTilt ( )
```

calculates how the robot is tilted. It is set, that when the robot is placed normally on a flat table, the result will be (0,0) Tilting the robot, so that the front leg is deeper than the other to results in an increasing degrees, tilting the front leg up will increase negativ degrees Tilting the robot to the right will increase the degrees until 180° (upside down), tilting it left will result in increasing negativ degrees (-1,-2,...,-180). On the top there is a jump of the values from 180->-180 and vice versa.

Precision is rounded to 1 deg steps

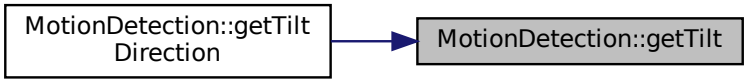
Attention

The results are only valid, if the robot is not moved in any way during the measurment, as the calculation is made by using the accelration values. If it's detected, that the robot is accelerated while measuring, the method will return max(int). Please note that the imu is pretty sensitiv, even walking next to the table may influcene the result.

Definition at line 78 of file MotionDetection.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



5.12.4.13 getTiltDirection()

```
Direction MotionDetection::getTiltDirection (
            uint tolerance = 10 )
```

Checks in which direction (Front, Left, Right, Back) the robot is tilted.

Attention

Does only work if the robot is not moving

Parameters

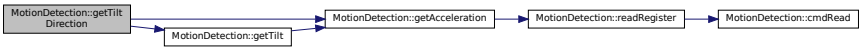
tolerance	(optional, default = 10) how many degrees can the robot be tilted, and still will be considerd as neutral.

Returns

Direction the direction in that the robot is tilted most. Front is onsiderd as the direction of driving. If robot is not tilted more than the tolerance in any direction, return is Neutral. If Robot is upside down, return is Flipped. If Robot is moved, return is Error

Definition at line 122 of file MotionDetection.cpp.

Here is the call graph for this function:



5.12.4.14 getWhoAmI()

```
int8_t MotionDetection::getWhoAmI (
            void  )
```

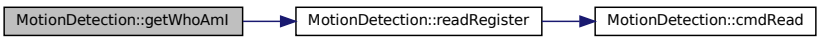
Returns the value of reading the whoAmI register When IMU working correctly, value should be 0x67.

Returns

the value of the whoami register of the ICM-42670

Definition at line 51 of file MotionDetection.cpp.

Here is the call graph for this function:

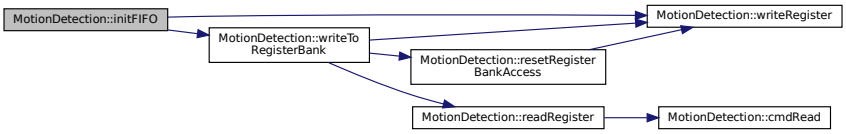


5.12.4.15 initFIFO()

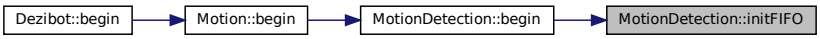
```
void MotionDetection::initFIFO ( )  [protected]
```

Definition at line 231 of file MotionDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.4.16 isShaken()

```
bool MotionDetection::isShaken (
            uint32_t threshold = defaultShakeThreshold,
            uint8_t axis = xAxis|yAxis|zAxis )
```

Detects if at the time of calling is shaken. Therefore the sum over all accelerationvalues is calculated and checked against threshold. If sum $>$ threshold a shake is detected, else not.

Parameters

threshold	(optional) the level of acceleration that must be reached to detect a shake
axis	(optional) select which axis should be used for detection. Possible values ar xAxis,yAxis,zAxis It's possible to combine multiple axis with the bitwise or Operator | For Example: to detect x and y axis: axis = xAxis|yAxis

Returns

true if a shake is detected, false else

Definition at line 55 of file MotionDetection.cpp.

Here is the call graph for this function:

5.12.4.17 readDoubleRegister()

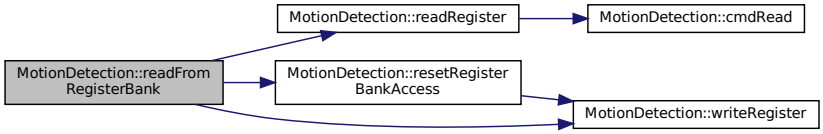
```
int16_t MotionDetection::readDoubleRegister (
            uint8_t lowerReg )  [protected]
```

5.12.4.18 readFromRegisterBank()

```
uint8_t MotionDetection::readFromRegisterBank (
            registerBank bank,
            uint8_t reg )  [protected]
```

Definition at line 173 of file MotionDetection.cpp.

Here is the call graph for this function:



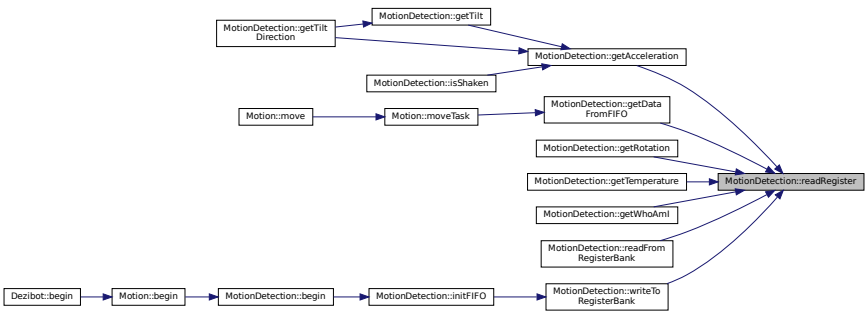
5.12.4.19 readRegister()

```
uint8_t MotionDetection::readRegister (
            uint8_t reg )  [protected]
```

Definition at line 162 of file MotionDetection.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

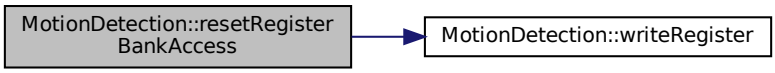


5.12.4.20 resetRegisterBankAccess()

```
void MotionDetection::resetRegisterBankAccess ( )  [protected]
```

Definition at line 224 of file MotionDetection.cpp.

Here is the call graph for this function:



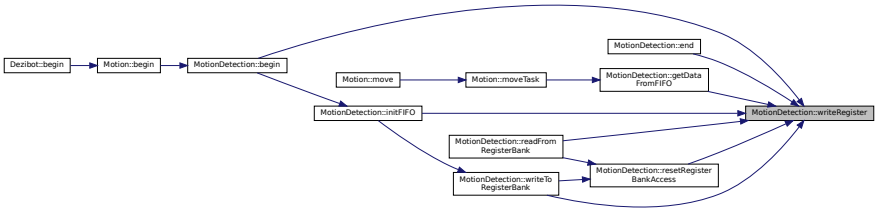
Here is the caller graph for this function:

5.12.4.21 writeRegister()

```
void MotionDetection::writeRegister (
            uint8_t reg,
            uint8_t value )  [protected]
```

Definition at line 279 of file MotionDetection.cpp.

Here is the caller graph for this function:

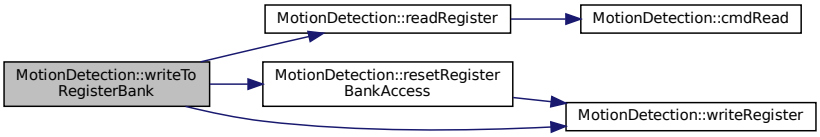


5.12.4.22 writeToRegisterBank()

```
void MotionDetection::writeToRegisterBank (
            registerBank bank,
            uint8_t reg,
            uint8_t value )  [protected]
```

Definition at line 194 of file MotionDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.12.5 Member Data Documentation

5.12.5.1 buf

```
int8_t* MotionDetection::buf = new int8_t[bufferLength]  [protected]
```

Definition at line 57 of file MotionDetection.h.

5.12.5.2 bufferLength

```
const uint MotionDetection::bufferLength = 64*16  [protected]
```

Definition at line 56 of file MotionDetection.h.

5.12.5.3 defaultShakeThreshold

```
const uint16_t MotionDetection::defaultShakeThreshold = 500  [static], [protected]
```

Definition at line 55 of file MotionDetection.h.

5.12.5.4 frequency

```
const uint MotionDetection::frequency = 24000000  [static], [protected]
```

Definition at line 54 of file MotionDetection.h.

5.12.5.5 gForceCalib

```
uint MotionDetection::gForceCalib = 4050  [protected]
```

Definition at line 74 of file MotionDetection.h.

5.12.5.6 handler

```
SPIClass* MotionDetection::handler = NULL  [protected]
```

Definition at line 72 of file MotionDetection.h.

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

- src/motionDetection/MotionDetection.h
- src/motionDetection/MotionDetection.cpp

5.13 Motor Class Reference

```
#include <Motion.h>
```

Public Member Functions

- Motor (uint8_t pin, ledc_timer_t timer, ledc_channel_t channel)
- void begin (void)

  Initializes the motor.
- void setSpeed (uint16_t duty)

  Set the Speed by changing the pwm. To avoid current peaks, a linear ramp-up is used.
- uint16_t getSpeed (void)

  returns the currently activ speed

Protected Attributes

- uint8_t pin
- ledc_timer_t timer
- ledc_channel_t channel
- uint16_t duty

5.13.1 Detailed Description

Definition at line 27 of file Motion.h.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 Motor()

```
Motor::Motor (
            uint8_t pin,
            ledc_timer_t timer,
            ledc_channel_t channel )
```

Definition at line 3 of file Motor.cpp.

5.13.3 Member Function Documentation

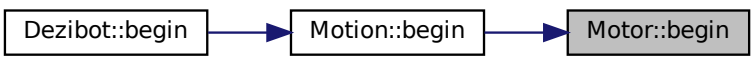
5.13.3.1 begin()

```
void Motor::begin (
            void  )
```

Initializes the motor.

Definition at line 10 of file Motor.cpp.

Here is the caller graph for this function:



5.13.3.2 getSpeed()

```
uint16_t Motor::getSpeed (
            void  )
```

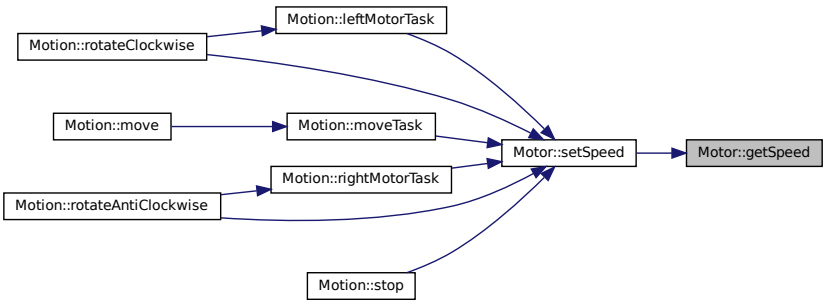
returns the currently activ speed

Returns

current speedvalue of the motor

Definition at line 46 of file Motor.cpp.

Here is the caller graph for this function:



5.13.3.3 setSpeed()

```
void Motor::setSpeed (
            uint16_t duty )
```

Set the Speed by changing the pwm. To avoid current peaks, a linear ramp-up is used.

Attention

it is requried at any time to use that method to access the motors or methods of the motionclass to avoid such peaks.

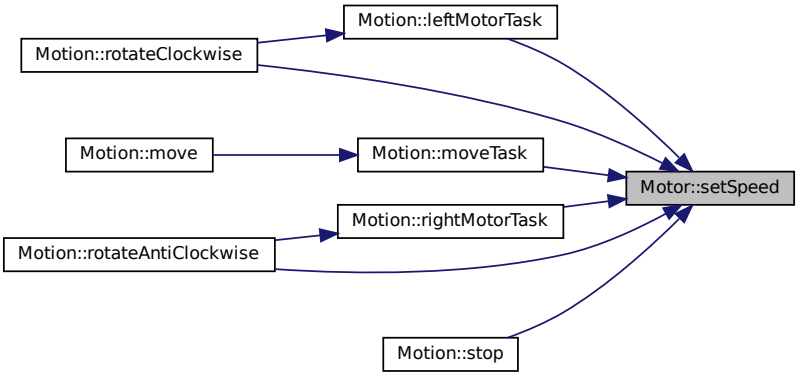
Parameters

duty	the duty cyle that should be set, can be between 0-8192

Definition at line 25 of file Motor.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



5.13.4 Member Data Documentation

5.13.4.1 channel

```
ledc_channel_t Motor::channel  [protected]
```

Definition at line 54 of file Motion.h.

5.13.4.2 duty

```
uint16_t Motor::duty  [protected]
```

Definition at line 56 of file Motion.h.

5.13.4.3 pin

```
uint8_t Motor::pin  [protected]
```

Definition at line 52 of file Motion.h.

5.13.4.4 timer

```
ledc_timer_t Motor::timer  [protected]
```

Definition at line 53 of file Motion.h.

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

- src/motion/Motion.h
- src/motion/Motor.cpp

5.14 MultiColorLight Class Reference

```
#include <MultiColorLight.h>
```

Public Member Functions

- MultiColorLight ()
- void begin (void)

  initialize the multicolor component
- void setLed (uint8_t index, uint32_t color)

  Set the specified led to the passed color.
- void setLed (leds leds, uint32_t color)

  Set the specified leds to the passed color value.
- void setLed (leds leds, uint8_t red, uint8_t green, uint8_t blue)

  Set the specified leds to the passed color value.
- void setTopLeds (uint32_t color)

  sets the two leds on the top of the robot to the specified color
- void setTopLeds (uint8_t red, uint8_t green, uint8_t blue)

  sets the two leds on the top of the robot to the specified color
- void blink (uint16_t amount, uint32_t color=0x00006400, leds leds=TOP, uint32_t interval=1000)

  Let LEDs blink, returns after all blinks were executed.
- void turnOffLed (leds leds=ALL)

  turn off the given leds
- uint32_t color (uint8_t r, uint8_t g, uint8_t b)

  wrapper to calulate the used colorformat from a rgb-value

Protected Attributes

- Adafruit_NeoPixel rgbLeds

Static Protected Attributes

- static const uint16_t ledAmount = 3
- static const int16_t ledPin = 48
- static const uint8_t maxBrightness = 150

5.14.1 Detailed Description

Definition at line 28 of file MultiColorLight.h.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 MultiColorLight()

```
MultiColorLight::MultiColorLight ( )
```

Definition at line 3 of file MultiColorLight.cpp.

5.14.3 Member Function Documentation

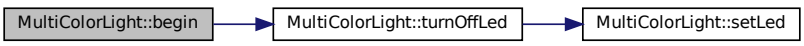
5.14.3.1 begin()

```
void MultiColorLight::begin (
            void  )
```

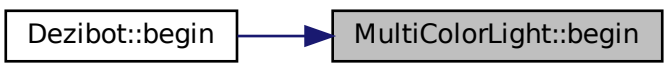
initialize the multicolor component

Definition at line 7 of file MultiColorLight.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.3.2 blink()

```
void MultiColorLight::blink (
            uint16_t amount,
            uint32_t color = 0x00006400,
            leds leds = TOP,
            uint32_t interval = 1000 )
```

Let LEDs blink, returns after all blinks were executed.

Parameters

amount	how often should the leds blink
color	A 32-bit unsigned integer representing the color in the format 0x00RRGGBB, where RR is the red component, GG is the green component, and BB is the blue component. Each color can range between 0 to 100 Defaults to blue
leds	which LEDs should blink, default is TOP
interval	how many miliseconds the led is on, defaults to 1s

Definition at line 57 of file MultiColorLight.cpp.

Here is the call graph for this function:

5.14.3.3 color()

```
uint32_t MultiColorLight::color (
            uint8_t r,
            uint8_t g,
            uint8_t b )
```

wrapper to calulate the used colorformat from a rgb-value

Parameters

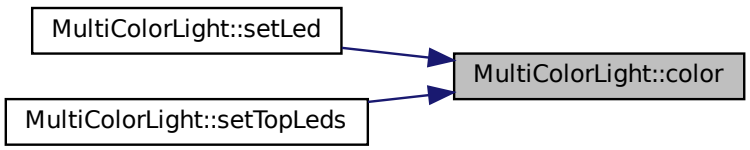
r	red (0-100)
g	green (0-100)
b	blue (0-100)

Returns

A 32-bit unsigned integer representing the color in the format 0x00RRGGBB, where RR is the red component, GG is the green component, and BB is the blue component.

Definition at line 88 of file MultiColorLight.cpp.

Here is the caller graph for this function:



5.14.3.4 setLed() [1/3]

```
void MultiColorLight::setLed (
            leds leds,
            uint32_t color )
```

Set the specified leds to the passed color value.

Parameters

leds	which leds should be updated
color	A 32-bit unsigned integer representing the color in the format 0x00RRGGBB, where RR is the red component, GG is the green component, and BB is the blue component. Each color can range between 0 to 100

Definition at line 21 of file MultiColorLight.cpp.

Here is the call graph for this function:

5.14.3.5 setLed() [2/3]

```
void MultiColorLight::setLed (
            leds leds,
            uint8_t red,
            uint8_t green,
            uint8_t blue )
```

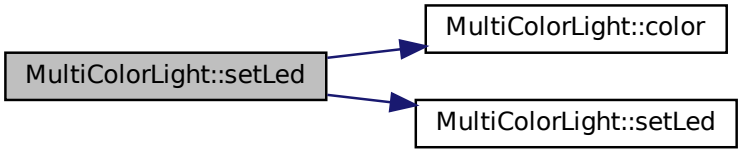
Set the specified leds to the passed color value.

Parameters

leds	which leds should be updated
red	brightness of red, is normalized in the function
green	brightness of green, is normalized in the function
blue	brightness of blue, is normalized in the function

Definition at line 44 of file MultiColorLight.cpp.

Here is the call graph for this function:



5.14.3.6 setLed() [3/3]

```
void MultiColorLight::setLed (
            uint8_t index,
            uint32_t color )
```

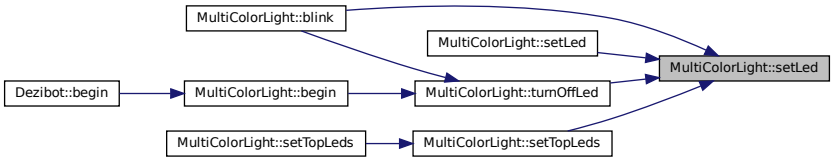
Set the specified led to the passed color.

Parameters

index	ranging from 0-2, 0: Right, 1: Left, 2: Bottom
color	A 32-bit unsigned integer representing the color in the format 0x00RRGGBB, where RR is the red component, GG is the green component, and BB is the blue component. Each color can range between 0 to 100

Definition at line 12 of file MultiColorLight.cpp.

Here is the caller graph for this function:



5.14.3.7 setTopLeds() [1/2]

```
void MultiColorLight::setTopLeds (
            uint32_t color )
```

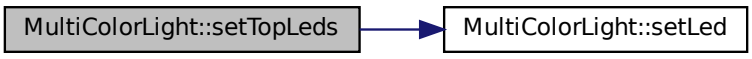
sets the two leds on the top of the robot to the specified color

Parameters

color	A 32-bit unsigned integer representing the color in the format 0x00RRGGBB, where RR is the red component, GG is the green component, and BB is the blue component. Each color can range between 0 to 100

Definition at line 49 of file MultiColorLight.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

5.14.3.8 setTopLeds() [2/2]

```
void MultiColorLight::setTopLeds (
            uint8_t red,
            uint8_t green,
            uint8_t blue )
```

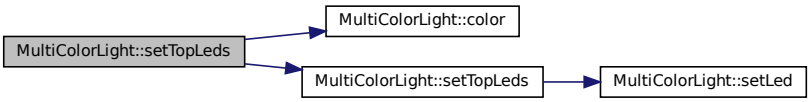
sets the two leds on the top of the robot to the specified color

Parameters

red	brightness of red, is normalized in the function
green	brightness of green, is normalized in the function
blue	brightness of blue, is normalized in the function

Definition at line 53 of file MultiColorLight.cpp.

Here is the call graph for this function:



5.14.3.9 turnOffLed()

```
void MultiColorLight::turnOffLed (
            leds leds = ALL )
```

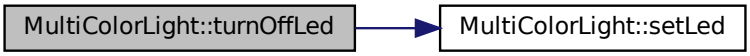
turn off the given leds

Parameters

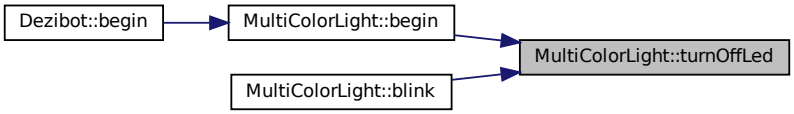
leds	which leds should be turned off, defaults to ALL

Definition at line 66 of file MultiColorLight.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.4 Member Data Documentation

5.14.4.1 ledAmount

```
const uint16_t MultiColorLight::ledAmount = 3  [static], [protected]
```

Definition at line 30 of file MultiColorLight.h.

5.14.4.2 ledPin

```
const int16_t MultiColorLight::ledPin = 48  [static], [protected]
```

Definition at line 31 of file MultiColorLight.h.

5.14.4.3 maxBrightness

```
const uint8_t MultiColorLight::maxBrightness = 150  [static], [protected]
```

Definition at line 32 of file MultiColorLight.h.

5.14.4.4 rgbLeds

```
Adafruit_NeoPixel MultiColorLight::rgbLeds  [protected]
```

Definition at line 33 of file MultiColorLight.h.

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

- src/multiColorLight/MultiColorLight.h
- src/multiColorLight/MultiColorLight.cpp

5.15 Orientation Struct Reference

```
#include <MotionDetection.h>
```

Public Attributes

- int xRotation
- int yRotation

5.15.1 Detailed Description

Definition at line 27 of file MotionDetection.h.

5.15.2 Member Data Documentation

5.15.2.1 xRotation

```
int Orientation::xRotation
```

Definition at line 28 of file MotionDetection.h.

5.15.2.2 yRotation

```
int Orientation::yRotation
```

Definition at line 29 of file MotionDetection.h.

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

- src/motionDetection/MotionDetection.h

5.16 VEML_CONFIG Struct Reference

`#include <ColorDetection.h>`

Public Attributes

- vemlMode mode
- bool enabled
- duration exposureTime

5.16.1 Detailed Description

Definition at line 37 of file ColorDetection.h.

5.16.2 Member Data Documentation

5.16.2.1 enabled

`bool VEML_CONFIG::enabled`

Definition at line 42 of file ColorDetection.h.

5.16.2.2 exposureTime

`duration VEML_CONFIG::exposureTime`

Definition at line 45 of file ColorDetection.h.

5.16.2.3 mode

`vemlMode VEML_CONFIG::mode`

Definition at line 39 of file ColorDetection.h.

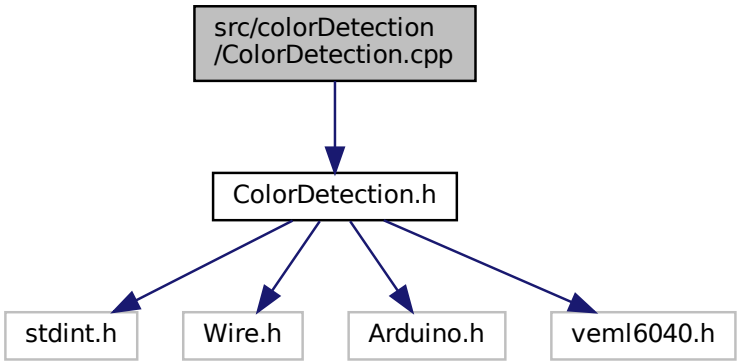
The documentation for this struct was generated from the following file:

- src/colorDetection/ColorDetection.h

Chapter 6

File Documentation

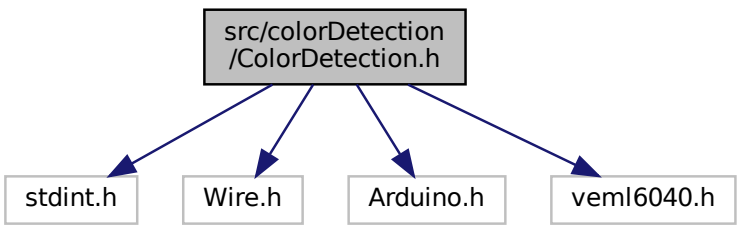
Include dependency graph for ColorDetection.cpp:



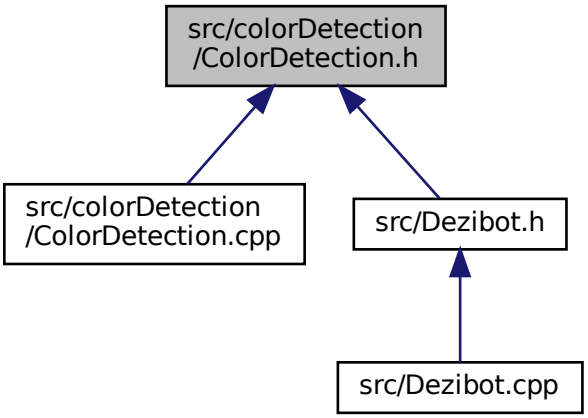
6.29 src/colorDetection/ColorDetection.h File Reference

```
#include <stdint.h>
#include <Wire.h>
#include <Arduino.h>
#include <veml6040.h>
```

Include dependency graph for ColorDetection.h:



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



Classes

- struct VEML_CONFIG
- class ColorDetection

Enumerations

- enum duration {
  MS40, MS80, MS160, MS320,
  MS640, MS1280 }
- enum vemlMode { AUTO, MANUAL }
- enum color { VEML_RED, VEML_GREEN, VEML_BLUE, VEML_WHITE }

6.29.1 Enumeration Type Documentation

6.29.1.1 color

`enum color`

Enumerator

VEML_RED	
VEML_GREEN	
VEML_BLUE	
VEML_WHITE	

Definition at line 48 of file ColorDetection.h.

6.29.1.2 duration

```
enum duration
```

Enumerator

MS40	
MS80	
MS160	
MS320	
MS640	
MS1280	

Definition at line 23 of file ColorDetection.h.

6.29.1.3 vemlMode

```
enum vemlMode
```

Enumerator

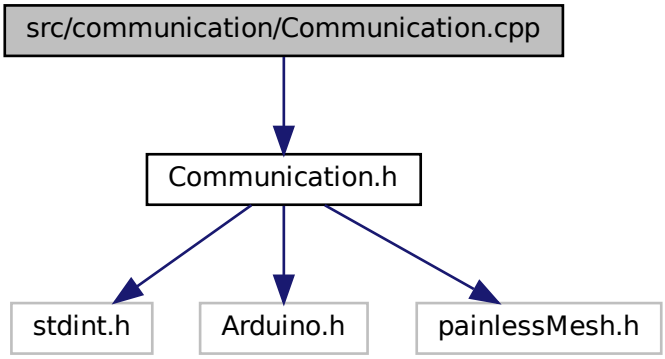
AUTO	
MANUAL	

Definition at line 32 of file ColorDetection.h.

6.30 src/communication/Communication.cpp File Reference

```
#include "Communication.h"
```

Include dependency graph for Communication.cpp:



Functions

- void newConnectionCallback (uint32_t nodeId)
- void changedConnectionCallback ()
- void nodeTimeAdjustedCallback (int32_t offset)
- void vTaskUpdate (void ∗pvParameters)

Variables

- Scheduler userScheduler
- painlessMesh mesh

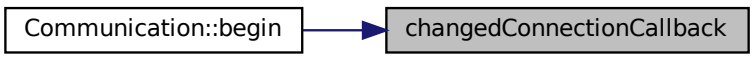
6.30.1 Function Documentation

6.30.1.1 changedConnectionCallback()

```
void changedConnectionCallback ( )
```

Definition at line 41 of file Communication.cpp.

Here is the caller graph for this function:



6.30.1.2 newConnectionCallback()

```
void newConnectionCallback (
            uint32_t nodeId )
```

Definition at line 36 of file Communication.cpp.

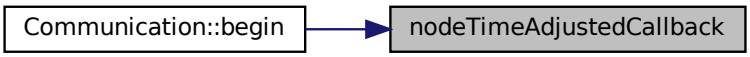
Here is the caller graph for this function:

6.30.1.3 nodeTimeAdjustedCallback()

```
void nodeTimeAdjustedCallback (
            int32_t offset )
```

Definition at line 46 of file Communication.cpp.

Here is the caller graph for this function:

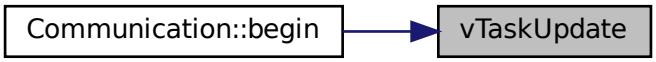


6.30.1.4 vTaskUpdate()

```
void vTaskUpdate (
            void * pvParameters )
```

Definition at line 51 of file Communication.cpp.

Here is the caller graph for this function:



6.30.2 Variable Documentation

6.30.2.1 mesh

```
painlessMesh mesh
```

Definition at line 4 of file Communication.cpp.

6.30.2.2 userScheduler

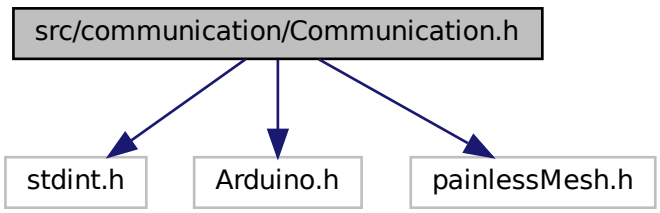
```
Scheduler userScheduler
```

Definition at line 3 of file Communication.cpp.

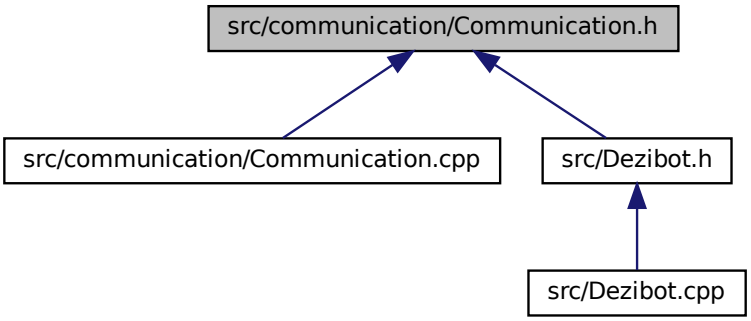
6.31 src/communication/Communication.h File Reference

```
#include <stdint.h>
#include <Arduino.h>
#include <painlessMesh.h>
```

Include dependency graph for Communication.h:



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



Classes

- class Communication

Macros

- #define MESH_PREFIX "DEZIBOT_MESH"
- #define MESH_PASSWORD "somethingSneaky"
- #define MESH_PORT 5555

6.31.1 Macro Definition Documentation

6.31.1.1 MESH_PASSWORD

```
#define MESH_PASSWORD "somethingSneaky"
```

Definition at line 9 of file Communication.h.

6.31.1.2 MESH_PORT

```
#define MESH_PORT 5555
```

Definition at line 10 of file Communication.h.

6.31.1.3 MESH_PREFIX

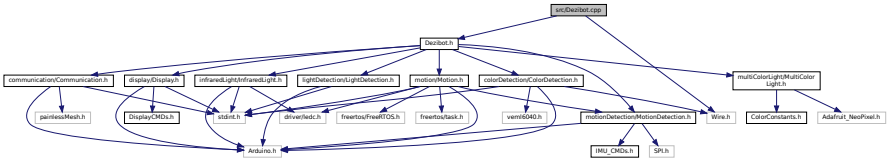
```
#define MESH_PREFIX "DEZIBOT_MESH"
```

Definition at line 8 of file Communication.h.

6.32 src/Dezibot.cpp File Reference

```
#include "Dezibot.h"
#include <Wire.h>
```

Include dependency graph for Dezibot.cpp:



Macros

- #define SDA_PIN 1
- #define SCL_PIN 2

6.32.1 Macro Definition Documentation

6.32.1.1 SCL_PIN

```
#define SCL_PIN 2
```

Definition at line 3 of file Dezibot.cpp.

6.32.1.2 SDA_PIN

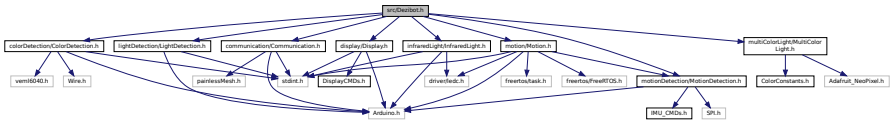
```
#define SDA_PIN 1
```

Definition at line 2 of file Dezibot.cpp.

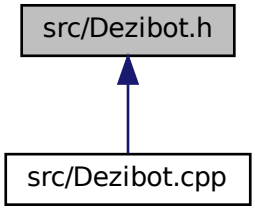
6.33 src/Dezibot.h File Reference

```
#include "motion/Motion.h"
#include "lightDetection/LightDetection.h"
#include "colorDetection/ColorDetection.h"
#include "multiColorLight/MultiColorLight.h"
#include "motionDetection/MotionDetection.h"
#include "infraredLight/InfraredLight.h"
#include "communication/Communication.h"
#include "display/Display.h"
```

Include dependency graph for Dezibot.h:



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



Classes

- class Dezibot

6.33.1 Detailed Description

Author

Hans Haupt, Jens Wagner, Anina Morgner, Anton Jacker, Saskia Dübener

Version

0.1

Date

2023-11-19

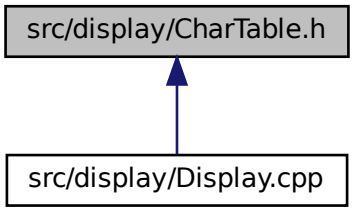
Copyright

Copyright (c) 2023

6.34 src/display/CharTable.h File Reference

LookUpTable for 8x8 Pixel Characters for an SSD1306 Display.

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



Variables

- const char font8x8_colwise [128][9]

  First index specifies the index, where index equals the ascii encoding unprintable characters are encode as all zero the first byte in an entry is the cmd_byte for SSD1306 and therefore not printed Encoding is colum wise, so first byte is the first column of the char and so on.

6.34.1 Detailed Description

LookUpTable for 8x8 Pixel Characters for an SSD1306 Display.

Author

Hans Haupt (hans.haupt@dezibot.de)

Version

0.1

Date

2024-05-24

Copyright

6.34.2 Variable Documentation

6.34.2.1 font8x8_colwise

```
const char font8x8_colwise[128][9]
```

First index specifies the index, where index equals the ascii encoding unprintable characters are encode as all zero the first byte in an entry is the cmd_byte for SSD1306 and therefore not printed Encoding is colum wise, so first byte is the first column of the char and so on.

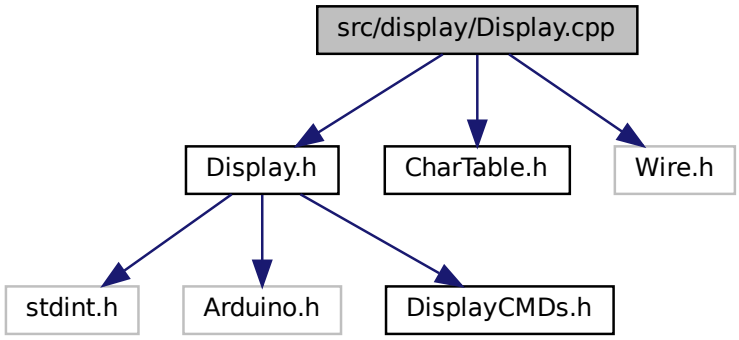
Definition at line 20 of file CharTable.h.

6.35 src/display/Display.cpp File Reference

Adds the ability to print to the display of the robot.

```
#include "Display.h"
#include "CharTable.h"
#include "Wire.h"
```

Include dependency graph for Display.cpp:



6.35.1 Detailed Description

Adds the ability to print to the display of the robot.

Author

Hans Haupt (hans.haupt@dezibot.de)

Version

0.1

Date

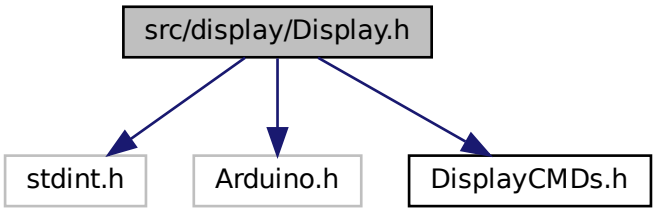
2024-06-05

Copyright

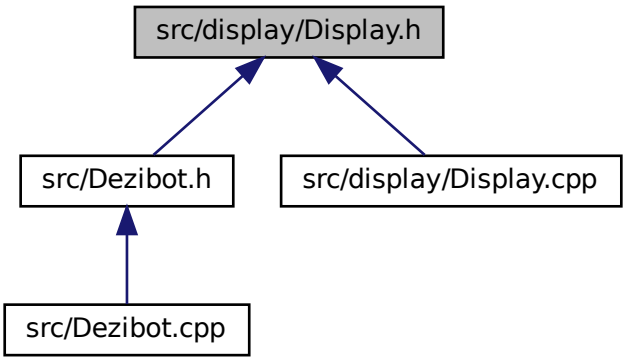
6.36 src/display/Display.h File Reference

```
#include <stdint.h>
#include <Arduino.h>
#include "DisplayCMDs.h"
```

Include dependency graph for Display.h:



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

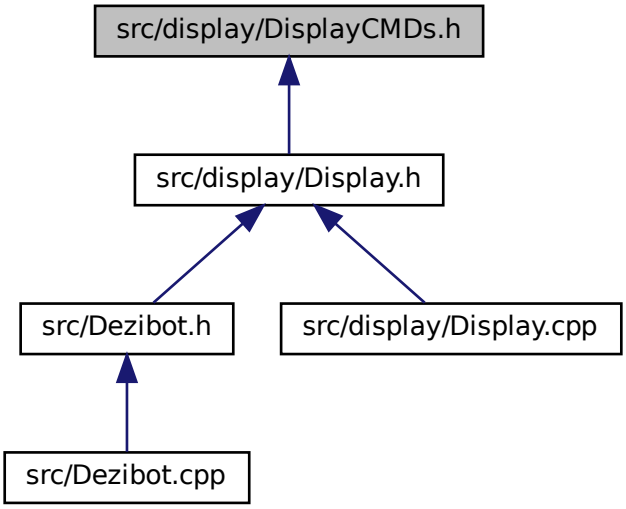


Classes

- class Display

6.37 src/display/DisplayCMDs.h File Reference

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



Macros

- #define cmd_byte 0x80
- #define data_byte 0x40
- #define muxRatio 0xa8
- #define setOffset 0xd3
- #define setStartLine 0x40
- #define setSegmentMap 0xa0
- #define setSegmentReMap 0xa1
- #define setComDirectionNormal 0xc0
- #define setComDirectionFlipped 0xc8
- #define setComHardwareConfig 0xda
- #define setContrast 0x81
- #define completeOn 0xa5
- #define stopCompleteOn 0xa4
- #define setNormalMode 0xa6
- #define setInverseMode 0xa7
- #define setOscFreq 0xd5
- #define setChargePump 0x8d
- #define activateDisplay 0xaf
- #define disableDisplay 0xae
- #define addressingMode 0x20
- #define colRange 0x21
- #define pageRange 0x22

6.37.1 Macro Definition Documentation

6.37.1.1 activateDisplay

```
#define activateDisplay 0xaf
```

Definition at line 18 of file DisplayCMDs.h.

6.37.1.2 addressingMode

```
#define addressingMode 0x20
```

Definition at line 20 of file DisplayCMDs.h.

6.37.1.3 cmd_byte

```
#define cmd_byte 0x80
```

Definition at line 1 of file DisplayCMDs.h.

6.37.1.4 colRange

```
#define colRange 0x21
```

Definition at line 21 of file DisplayCMDs.h.

6.37.1.5 completeOn

```
#define completeOn 0xa5
```

Definition at line 12 of file DisplayCMDs.h.

6.37.1.6 data_byte

```
#define data_byte 0x40
```

Definition at line 2 of file DisplayCMDs.h.

6.37.1.7 disableDisplay

```
#define disableDisplay 0xae
```

Definition at line 19 of file DisplayCMDs.h.

6.37.1.8 muxRatio

```
#define muxRatio 0xa8
```

Definition at line 3 of file DisplayCMDs.h.

6.37.1.9 pageRange

```
#define pageRange 0x22
```

Definition at line 22 of file DisplayCMDs.h.

6.37.1.10 setChargePump

```
#define setChargePump 0x8d
```

Definition at line 17 of file DisplayCMDs.h.

6.37.1.11 setComDirectionFlipped

```
#define setComDirectionFlipped 0xc8
```

Definition at line 9 of file DisplayCMDs.h.

6.37.1.12 setComDirectionNormal

```
#define setComDirectionNormal 0xc0
```

Definition at line 8 of file DisplayCMDs.h.

6.37.1.13 setComHardwareConfig

```
#define setComHardwareConfig 0xda
```

Definition at line 10 of file DisplayCMDs.h.

6.37.1.14 setContrast

```
#define setContrast 0x81
```

Definition at line 11 of file DisplayCMDs.h.

6.37.1.15 setInverseMode

```
#define setInverseMode 0xa7
```

Definition at line 15 of file DisplayCMDs.h.

6.37.1.16 setNormalMode

```
#define setNormalMode 0xa6
```

Definition at line 14 of file DisplayCMDs.h.

6.37.1.17 setOffset

```
#define setOffset 0xd3
```

Definition at line 4 of file DisplayCMDs.h.

6.37.1.18 setOscFreq

```
#define setOscFreq 0xd5
```

Definition at line 16 of file DisplayCMDs.h.

6.37.1.19 setSegmentMap

```
#define setSegmentMap 0xa0
```

Definition at line 6 of file DisplayCMDs.h.

6.37.1.20 setSegmentReMap

```
#define setSegmentReMap 0xa1
```

Definition at line 7 of file DisplayCMDs.h.

6.37.1.21 setStartLine

```
#define setStartLine 0x40
```

Definition at line 5 of file DisplayCMDs.h.

6.37.1.22 stopCompleteOn

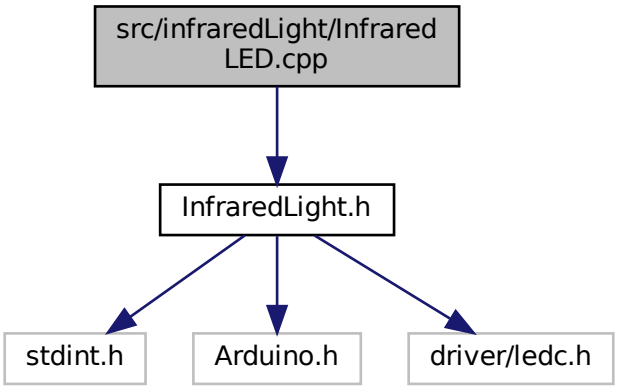
```
#define stopCompleteOn 0xa4
```

Definition at line 13 of file DisplayCMDs.h.

6.38 src/infraredLight/InfraredLED.cpp File Reference

```
#include "InfraredLight.h"
```

Include dependency graph for InfraredLED.cpp:



Macros

- #define pwmSpeedMode LEDC_LOW_SPEED_MODE

6.38.1 Macro Definition Documentation

6.38.1.1 pwmSpeedMode

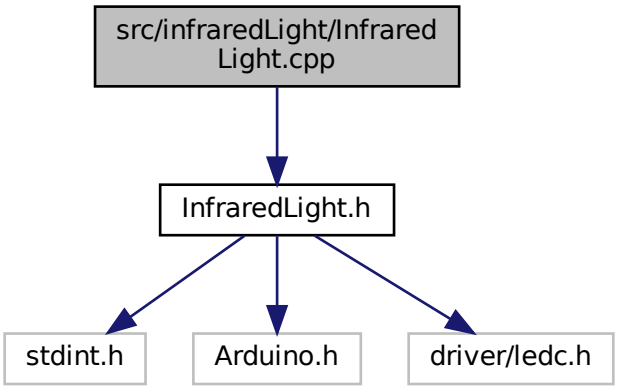
```
#define pwmSpeedMode LEDC_LOW_SPEED_MODE
```

Definition at line 3 of file InfraredLED.cpp.

6.39 src/infraredLight/InfraredLight.cpp File Reference

```
#include "InfraredLight.h"
```

Include dependency graph for InfraredLight.cpp:

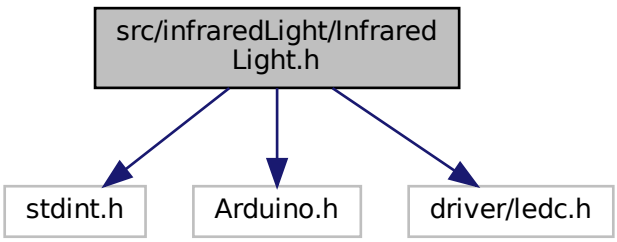


6.40 src/infraredLight/InfraredLight.h File Reference

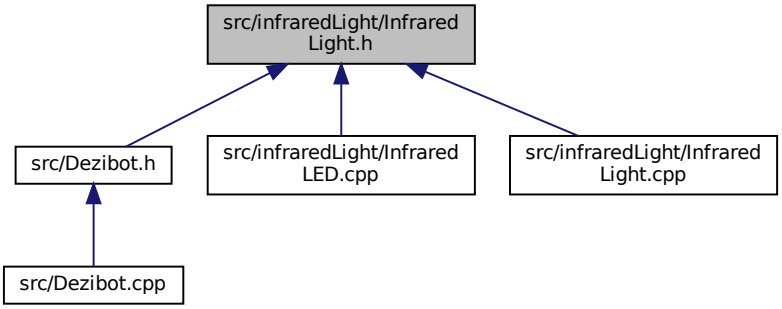
Adds the ability to print to the display of the robot.

```
#include <stdint.h>
#include <Arduino.h>
#include "driver/ledc.h"
```

Include dependency graph for InfraredLight.h:



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



Classes

- class InfraredLED
- class InfraredLight

6.40.1 Detailed Description

Adds the ability to print to the display of the robot.

Provides basic controls for the infrared LEDs of the robot.

Author

Hans Haupt (hans.haupt@dezibot.de)

Version

0.1

Date

2024-05-24

Copyright

Copyright (c) 2024

Author

Hans Haupt (hans.haupt@dezibot.de)

Version

0.1

Date

2024-04-27

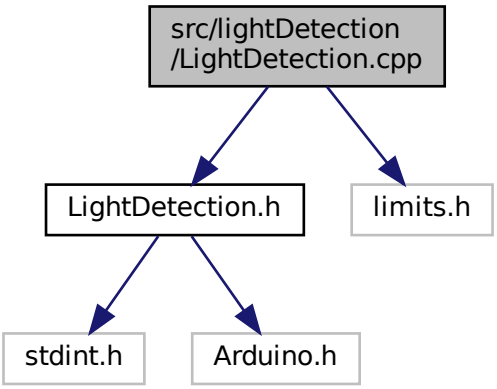
Copyright

Copyright (c) 2024

6.41 src/lightDetection/LightDetection.cpp File Reference

```
#include "LightDetection.h"
#include <limits.h>
```

Include dependency graph for LightDetection.cpp:

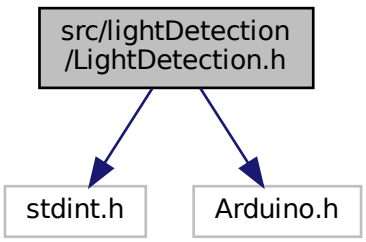


6.42 src/lightDetection/LightDetection.h File Reference

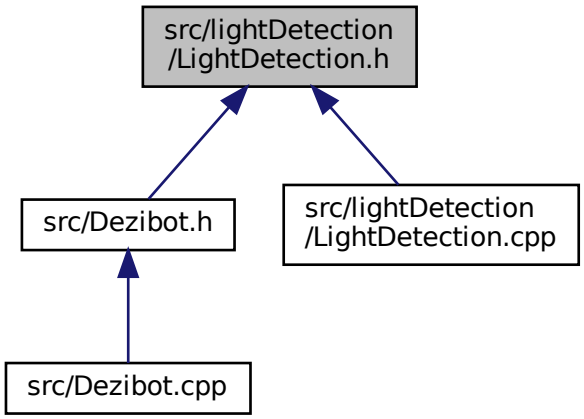
Class for Reading the values of the different Phototransistors, both IR, and DaylightSensors are supported.

```
#include <stdint.h>
#include <Arduino.h>
```

Include dependency graph for LightDetection.h:



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



Classes

- struct averageMeasurement
- class LightDetection

Enumerations

- enum photoTransistors { IR_LEFT, IR_RIGHT, IR_FRONT, IR_BACK, DL_FRONT, DL_BOTTOM }
- enum ptType { IR, DAYLIGHT }

6.42.1 Detailed Description

Class for Reading the values of the different Phototransistors, both IR, and DaylightSensors are supported.

Author

Hans Haupt (hans.haupt@dezibot.de)

Version

0.1

Date

2024-04-26

Copyright

6.42.2 Enumeration Type Documentation

6.42.2.1 photoTransistors

```
enum photoTransistors
```

Enumerator

IR_LEFT	
IR_RIGHT	
IR_FRONT	
IR_BACK	
DL_FRONT	
DL_BOTTOM	

Definition at line 19 of file LightDetection.h.

6.42.2.2 ptType

```
enum ptType
```

Enumerator

IR	
DAYLIGHT	

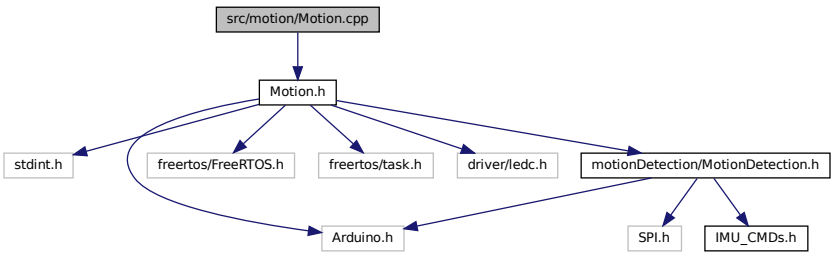
Definition at line 36 of file LightDetection.h.

6.43 src/motion/Motion.cpp File Reference

Implementation of the Motion class.

```
#include "Motion.h"
```

Include dependency graph for Motion.cpp:



6.43.1 Detailed Description

Implementation of the Motion class.

Author

Jonathan Schulze, Nick Hübenthal, Hans Haupt

Version

0.2

Date

2023-12-13

Copyright

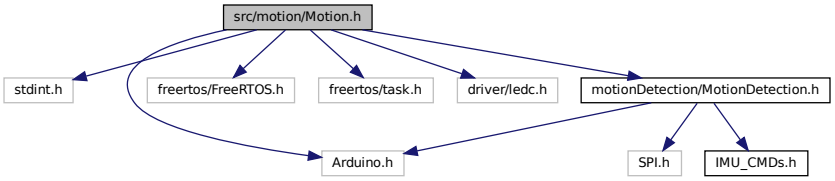
6.44 src/motion/Motion.h File Reference

This component controls the ability to rotate and change position.

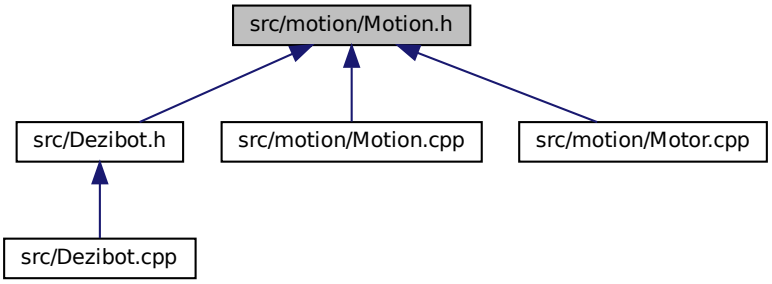
```
#include <stdint.h>
#include <Arduino.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include "driver/ledc.h"
```

```
#include "motionDetection/MotionDetection.h"
```

Include dependency graph for Motion.h:



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



Classes

- class Motor
- class Motion

Macros

- #define LEDC_MODE LEDC_LOW_SPEED_MODE
- #define TIMER LEDC_TIMER_2
- #define CHANNEL_LEFT LEDC_CHANNEL_3
- #define CHANNEL_RIGHT LEDC_CHANNEL_4
- #define DUTY_RES LEDC_TIMER_13_BIT
- #define FREQUENCY (5000)
- #define DEFAULT_BASE_VALUE 3900

6.44.1 Detailed Description

This component controls the ability to rotate and change position.

Author

Jonathan Schulze, Nick Hübenthal, Hans Haupt

Version

0.2

Date

2023-12-13

Copyright

6.44.2 Macro Definition Documentation

6.44.2.1 CHANNEL_LEFT

```
#define CHANNEL_LEFT LEDC_CHANNEL_3
```

Definition at line 22 of file Motion.h.

6.44.2.2 CHANNEL_RIGHT

```
#define CHANNEL_RIGHT LEDC_CHANNEL_4
```

Definition at line 23 of file Motion.h.

6.44.2.3 DEFAULT_BASE_VALUE

```
#define DEFAULT_BASE_VALUE 3900
```

Definition at line 26 of file Motion.h.

6.44.2.4 DUTY_RES

```
#define DUTY_RES LEDC_TIMER_13_BIT
```

Definition at line 24 of file Motion.h.

6.44.2.5 FREQUENCY

```
#define FREQUENCY (5000)
```

Definition at line 25 of file Motion.h.

6.44.2.6 LEDC_MODE

```
#define LEDC_MODE LEDC_LOW_SPEED_MODE
```

Definition at line 20 of file Motion.h.

6.44.2.7 TIMER

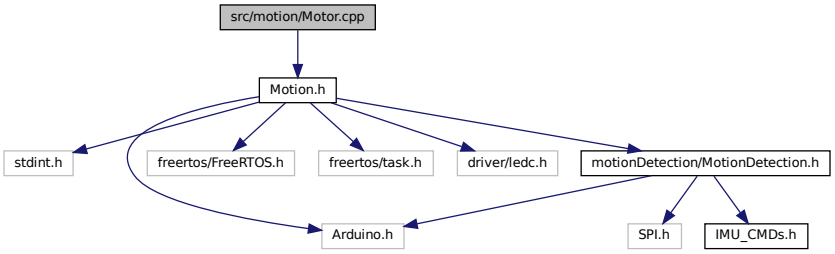
```
#define TIMER LEDC_TIMER_2
```

Definition at line 21 of file Motion.h.

6.45 src/motion/Motor.cpp File Reference

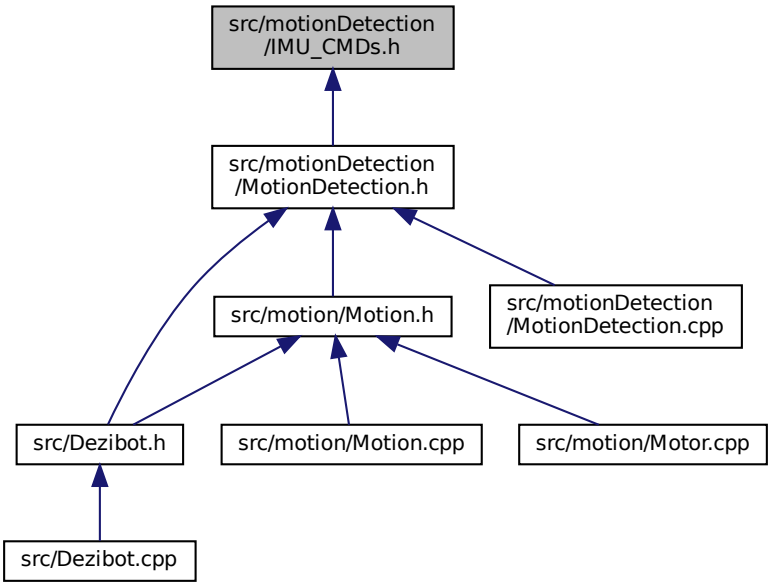
```
#include "Motion.h"
```

Include dependency graph for Motor.cpp:



6.46 src/motionDetection/IMU_CMDs.h File Reference

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



Macros

- #define CMD_READ 0x80
- #define CMD_WRITE 0x00
- #define ADDR_MASK 0x7F
- #define MCLK_RDY 0x00
- #define REG_TEMP_LOW 0x0A
- #define REG_TEMP_HIGH 0X09
- #define ACCEL_DATA_X_HIGH 0x0B
- #define ACCEL_DATA_X_LOW 0x0C
- #define ACCEL_DATA_Y_HIGH 0x0D
- #define ACCEL_DATA_Y_LOW 0x0E
- #define ACCEL_DATA_Z_HIGH 0x0F
- #define ACCEL_DATA_Z_LOW 0x10
- #define GYRO_DATA_X_HIGH 0x11
- #define GYRO_DATA_X_LOW 0x12
- #define GYRO_DATA_Y_HIGH 0x13
- #define GYRO_DATA_Y_LOW 0x14
- #define GYRO_DATA_Z_HIGH 0x15
- #define GYRO_DATA_Z_LOW 0x16
- #define PWR_MGMT0 0x1F
- #define WHO_AM_I 0x75
- #define INTF_CONFIG0 0x35
- #define BLK_SEL_W 0x79
- #define BLK_SEL_R 0x7C

- #define MADDR_W 0x7A
- #define MADDR_R 0x7D
- #define M_W 0x7B
- #define M_R 0x7E
- #define FIFO_COUNTH 0x3D
- #define FIFO_COUNTL 0x3E
- #define FIFO_DATA 0x3F
- #define FIFO_CONFIG1 0x28
- #define FIFO_CONFIG2 0x29
- #define FIFO_CONFIG5 0x01
- #define TMST_CONFIG1 0x00

6.46.1 Macro Definition Documentation

6.46.1.1 ACCEL_DATA_X_HIGH

```
#define ACCEL_DATA_X_HIGH 0x0B
```

Definition at line 14 of file IMU_CMDs.h.

6.46.1.2 ACCEL_DATA_X_LOW

```
#define ACCEL_DATA_X_LOW 0x0C
```

Definition at line 15 of file IMU_CMDs.h.

6.46.1.3 ACCEL_DATA_Y_HIGH

```
#define ACCEL_DATA_Y_HIGH 0x0D
```

Definition at line 16 of file IMU_CMDs.h.

6.46.1.4 ACCEL_DATA_Y_LOW

```
#define ACCEL_DATA_Y_LOW 0x0E
```

Definition at line 17 of file IMU_CMDs.h.

6.46.1.5 ACCEL_DATA_Z_HIGH

```
#define ACCEL_DATA_Z_HIGH 0x0F
```

Definition at line 18 of file IMU_CMDs.h.

6.46.1.6 ACCEL_DATA_Z_LOW

```
#define ACCEL_DATA_Z_LOW 0x10
```

Definition at line 19 of file IMU_CMDs.h.

6.46.1.7 ADDR_MASK

```
#define ADDR_MASK 0x7F
```

Definition at line 6 of file IMU_CMDs.h.

6.46.1.8 BLK_SEL_R

```
#define BLK_SEL_R 0x7C
```

Definition at line 34 of file IMU_CMDs.h.

6.46.1.9 BLK_SEL_W

```
#define BLK_SEL_W 0x79
```

Definition at line 33 of file IMU_CMDs.h.

6.46.1.10 CMD_READ

```
#define CMD_READ 0x80
```

Definition at line 4 of file IMU_CMDs.h.

6.46.1.11 CMD_WRITE

```
#define CMD_WRITE 0x00
```

Definition at line 5 of file IMU_CMDs.h.

6.46.1.12 FIFO_CONFIG1

```
#define FIFO_CONFIG1 0x28
```

Definition at line 43 of file IMU_CMDs.h.

6.46.1.13 FIFO_CONFIG2

```
#define FIFO_CONFIG2 0x29
```

Definition at line 44 of file IMU_CMDs.h.

6.46.1.14 FIFO_CONFIG5

```
#define FIFO_CONFIG5 0x01
```

Definition at line 47 of file IMU_CMDs.h.

6.46.1.15 FIFO_COUNTH

```
#define FIFO_COUNTH 0x3D
```

Definition at line 40 of file IMU_CMDs.h.

6.46.1.16 FIFO_COUNTL

```
#define FIFO_COUNTL 0x3E
```

Definition at line 41 of file IMU_CMDs.h.

6.46.1.17 FIFO_DATA

```
#define FIFO_DATA 0x3F
```

Definition at line 42 of file IMU_CMDs.h.

6.46.1.18 GYRO_DATA_X_HIGH

```
#define GYRO_DATA_X_HIGH 0x11
```

Definition at line 21 of file IMU_CMDs.h.

6.46.1.19 GYRO_DATA_X_LOW

```
#define GYRO_DATA_X_LOW 0x12
```

Definition at line 22 of file IMU_CMDs.h.

6.46.1.20 GYRO_DATA_Y_HIGH

```
#define GYRO_DATA_Y_HIGH 0x13
```

Definition at line 23 of file IMU_CMDs.h.

6.46.1.21 GYRO_DATA_Y_LOW

```
#define GYRO_DATA_Y_LOW 0x14
```

Definition at line 24 of file IMU_CMDs.h.

6.46.1.22 GYRO_DATA_Z_HIGH

```
#define GYRO_DATA_Z_HIGH 0x15
```

Definition at line 25 of file IMU_CMDs.h.

6.46.1.23 GYRO_DATA_Z_LOW

```
#define GYRO_DATA_Z_LOW 0x16
```

Definition at line 26 of file IMU_CMDs.h.

6.46.1.24 INTF_CONFIG0

```
#define INTF_CONFIG0 0x35
```

Definition at line 31 of file IMU_CMDs.h.

6.46.1.25 M_R

```
#define M_R 0x7E
```

Definition at line 38 of file IMU_CMDs.h.

6.46.1.26 M_W

```
#define M_W 0x7B
```

Definition at line 37 of file IMU_CMDs.h.

6.46.1.27 MADDR_R

```
#define MADDR_R 0x7D
```

Definition at line 36 of file IMU_CMDs.h.

6.46.1.28 MADDR_W

```
#define MADDR_W 0x7A
```

Definition at line 35 of file IMU_CMDs.h.

6.46.1.29 MCLK_RDY

```
#define MCLK_RDY 0x00
```

Definition at line 9 of file IMU_CMDs.h.

6.46.1.30 PWR_MGMT0

```
#define PWR_MGMT0 0x1F
```

Definition at line 28 of file IMU_CMDs.h.

6.46.1.31 REG_TEMP_HIGH

```
#define REG_TEMP_HIGH 0X09
```

Definition at line 12 of file IMU_CMDs.h.

6.46.1.32 REG_TEMP_LOW

```
#define REG_TEMP_LOW 0x0A
```

Definition at line 11 of file IMU_CMDs.h.

6.46.1.33 TMST_CONFIG1

```
#define TMST_CONFIG1 0x00
```

Definition at line 48 of file IMU_CMDs.h.

6.46.1.34 WHO_AM_I

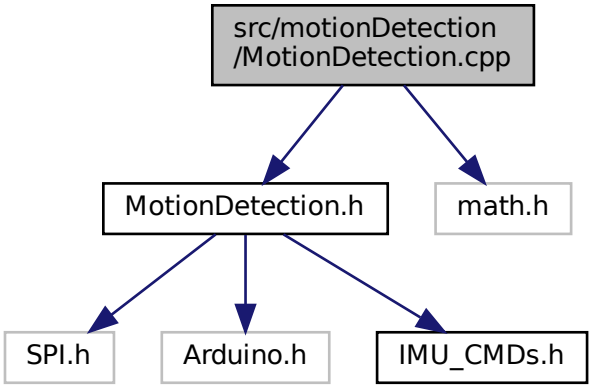
```
#define WHO_AM_I 0x75
```

Definition at line 29 of file IMU_CMDs.h.

6.47 src/motionDetection/MotionDetection.cpp File Reference

```
#include "MotionDetection.h"
#include <math.h>
```

Include dependency graph for MotionDetection.cpp:

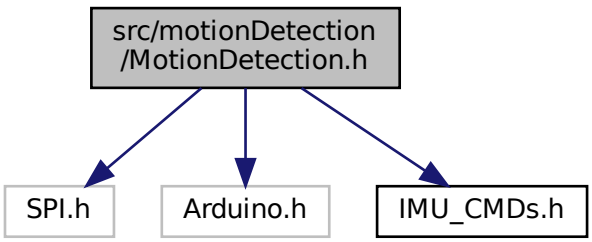


6.48 src/motionDetection/MotionDetection.h File Reference

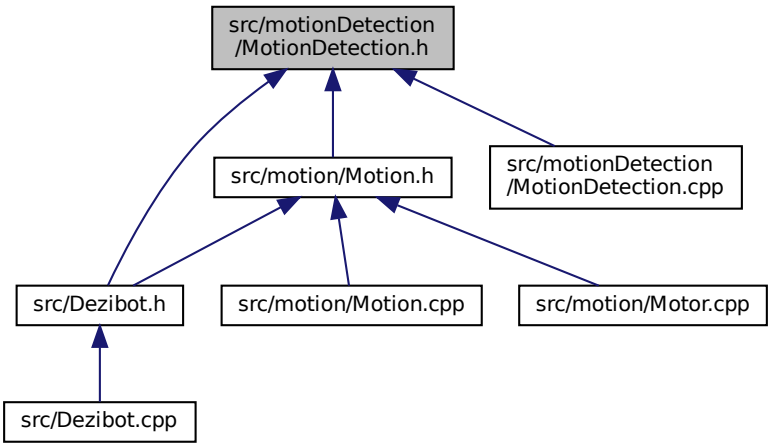
This component controls the IMU (Accelerometer & Gyroscope) ICM-42670-P.

```
#include <SPI.h>
#include <Arduino.h>
#include "IMU_CMDs.h"
```

Include dependency graph for MotionDetection.h:



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



Classes

- struct IMUResult
- struct Orientation
- struct FIFO_Package
- class MotionDetection

Enumerations

- enum Axis { xAxis = 0x01, yAxis = 0x02, zAxis = 0x04 }
- enum Direction {
  Front, Left, Right, Back,
  Neutral, Flipped, Error }

6.48.1 Detailed Description

This component controls the IMU (Accelerometer & Gyroscope) ICM-42670-P.

Author

Hans Haupt

Version

0.1

Date

2023-12-15

Copyright

6.48.2 Enumeration Type Documentation

6.48.2.1 Axis

```
enum Axis
```

Enumerator

xAxis	
yAxis	
zAxis	

Definition at line 21 of file MotionDetection.h.

6.48.2.2 Direction

```
enum Direction
```

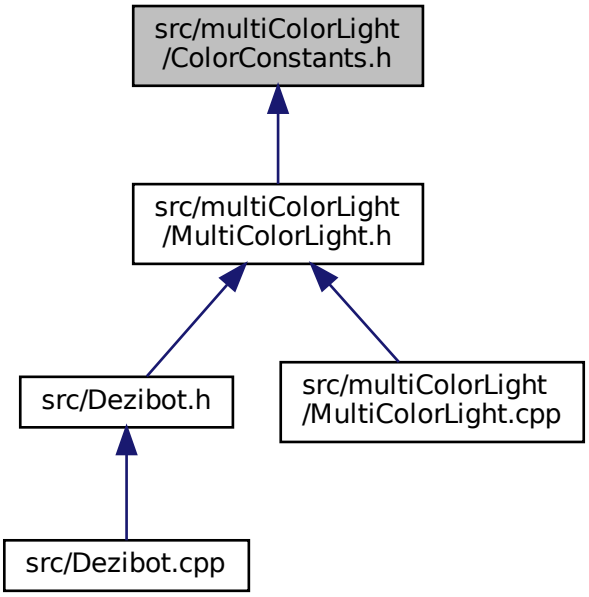
Enumerator

Front	
Left	
Right	
Back	
Neutral	
Flipped	
Error	

Definition at line 32 of file MotionDetection.h.

6.49 src/multiColorLight/ColorConstants.h File Reference

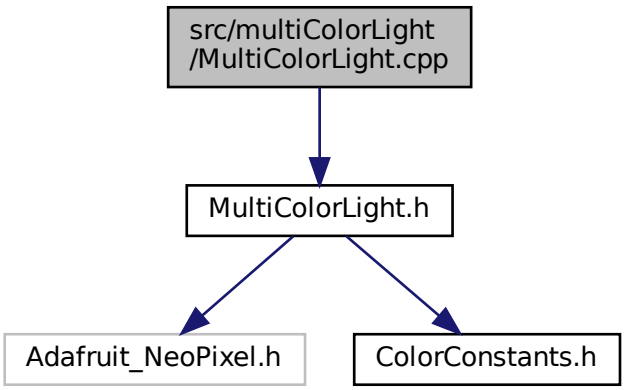
This graph shows which files directly or indirectly include this file:



6.50 src/multiColorLight/MultiColorLight.cpp File Reference

```
#include "MultiColorLight.h"
```

Include dependency graph for MultiColorLight.cpp:

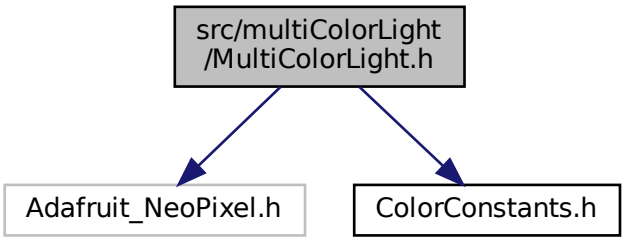


6.51 src/multiColorLight/MultiColorLight.h File Reference

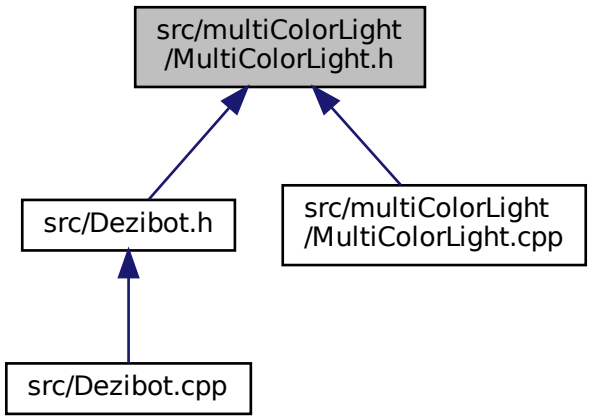
This component controls the ability to show multicolored light, using the RGB-LEDs.

```
#include <Adafruit_NeoPixel.h>
#include "ColorConstants.h"
```

Include dependency graph for MultiColorLight.h:



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



Classes

- class MultiColorLight

Enumerations

- enum leds {
  TOP_LEFT, TOP_RIGHT, BOTTOM, TOP,
  ALL }

  Describes combinations of leds on the Dezibot. With the Robot in Front of you, when the robot drives away from you, the left LED is TOP_LEFT.

6.51.1 Detailed Description

This component controls the ability to show multicolored light, using the RGB-LEDs.

Author

Saskia Duebener, Hans Haupt

Version

0.2

Date

2023-11-25

Copyright

Copyright (c) 2023

6.51.2 Enumeration Type Documentation

6.51.2.1 leds

`enum leds`

Describes combinations of leds on the Dezibot. With the Robot in Front of you, when the robot drives away from you, the left LED is TOP_LEFT.

Enumerator

TOP_LEFT	
TOP_RIGHT	
BOTTOM	
TOP	
ALL	

Definition at line 20 of file MultiColorLight.h.

Index