

Dezibot 4

Generated by Doxygen 1.8.17



---

<b>1 Dezibot4 Lib</b>	<b>1</b>
1.1 Links to external documentation	1
1.2 Link to Software	1
1.3 Introduction	1
1.4 Code Conventions	1
1.4.1 Don't pass reference	1
1.4.2 Naming	2
1.4.3 Bytestream	2
1.4.4 Components	2
1.5 Design Paradigm	2
1.5.1 Part Instances	2
1.5.2 Abstractions	2
1.6 Contributing	2
1.6.1 Branching	3
1.6.2 Mergerequests	3
1.6.3 Commitmessages	3
1.6.4 Language	3
1.6.5 Documentation	3
1.6.5.1 .h Files	3
1.6.5.2 Methods	4
1.6.5.3 Arduino Settings	4
1.6.5.4 Start from Scratch	4
1.7 Third-Party Licenses	4
<b>2 Dezibot4 library</b>	<b>5</b>
<b>3 Class Index</b>	<b>7</b>
3.1 Class List	7
<b>4 File Index</b>	<b>9</b>
4.1 File List	9
<b>5 Class Documentation</b>	<b>11</b>
5.1 averageMeasurement Struct Reference	11
5.1.1 Detailed Description	11
5.1.2 Member Data Documentation	11
5.1.2.1 done	11
5.1.2.2 measurementAmount	12
5.1.2.3 result	12
5.1.2.4 sensor	12
5.1.2.5 timeBetween	12
5.2 ColorDetection Class Reference	12
5.2.1 Detailed Description	13
5.2.2 Member Function Documentation	13

---

---

5.2.2.1 beginAutoMode()	13
5.2.2.2 configure()	13
5.2.2.3 getAmbientLight()	14
5.2.2.4 getColorValue()	14
5.2.3 Member Data Documentation	15
5.2.3.1 rgbwSensor	15
5.3 Communication Class Reference	15
5.3.1 Detailed Description	15
5.3.2 Member Function Documentation	15
5.3.2.1 begin()	16
5.3.2.2 onReceive()	16
5.3.2.3 sendMessage()	16
5.3.2.4 setGroupNumber()	17
5.4 Dezipot Class Reference	17
5.4.1 Detailed Description	18
5.4.2 Constructor & Destructor Documentation	18
5.4.2.1 Dezipot()	18
5.4.3 Member Function Documentation	18
5.4.3.1 begin()	18
5.4.4 Member Data Documentation	18
5.4.4.1 colorDetection	18
5.4.4.2 communication	19
5.4.4.3 display	19
5.4.4.4 infraredLight	19
5.4.4.5 lightDetection	19
5.4.4.6 motion	19
5.4.4.7 multiColorLight	19
5.5 Display Class Reference	20
5.5.1 Detailed Description	21
5.5.2 Member Function Documentation	21
5.5.2.1 begin()	21
5.5.2.2 clear()	22
5.5.2.3 flipOrientation()	22
5.5.2.4 invertColor()	23
5.5.2.5 print() [1/3]	23
5.5.2.6 print() [2/3]	24
5.5.2.7 print() [3/3]	24
5.5.2.8 println() [1/3]	25
5.5.2.9 println() [2/3]	26
5.5.2.10 println() [3/3]	26
5.5.2.11 sendDisplayCMD()	27
5.5.2.12 stringToCharArray()	27

---

5.5.2.13 updateLine() . . . . .	27
5.5.3 Member Data Documentation . . . . .	28
5.5.3.1 charsOnCurrLine . . . . .	28
5.5.3.2 colorInverted . . . . .	28
5.5.3.3 currLine . . . . .	28
5.5.3.4 orientationFlipped . . . . .	29
5.6 FIFO_Package Struct Reference . . . . .	29
5.6.1 Detailed Description . . . . .	29
5.6.2 Member Data Documentation . . . . .	29
5.6.2.1 accel . . . . .	30
5.6.2.2 gyro . . . . .	30
5.6.2.3 header . . . . .	30
5.6.2.4 temperature . . . . .	30
5.6.2.5 timestamp . . . . .	30
5.7 IMUResult Struct Reference . . . . .	30
5.7.1 Detailed Description . . . . .	31
5.7.2 Member Data Documentation . . . . .	31
5.7.2.1 x . . . . .	31
5.7.2.2 y . . . . .	31
5.7.2.3 z . . . . .	31
5.8 InfraredLED Class Reference . . . . .	31
5.8.1 Detailed Description . . . . .	32
5.8.2 Constructor & Destructor Documentation . . . . .	32
5.8.2.1 InfraredLED() . . . . .	32
5.8.3 Member Function Documentation . . . . .	32
5.8.3.1 begin() . . . . .	33
5.8.3.2 sendFrequency() . . . . .	33
5.8.3.3 setState() . . . . .	33
5.8.3.4 turnOff() . . . . .	34
5.8.3.5 turnOn() . . . . .	34
5.8.4 Member Data Documentation . . . . .	35
5.8.4.1 channel . . . . .	35
5.8.4.2 ledPin . . . . .	35
5.8.4.3 pwmChannel . . . . .	35
5.8.4.4 pwmTimer . . . . .	35
5.8.4.5 timer . . . . .	36
5.9 InfraredLight Class Reference . . . . .	36
5.9.1 Detailed Description . . . . .	37
5.9.2 Member Function Documentation . . . . .	37
5.9.2.1 begin() . . . . .	37
5.9.3 Member Data Documentation . . . . .	37
5.9.3.1 bottom . . . . .	37

---

5.9.3.2 front	38
5.9.3.3 IRBottomPin	38
5.9.3.4 IRFrontPin	38
5.10 LightDetection Class Reference	38
5.10.1 Detailed Description	39
5.10.2 Member Function Documentation	39
5.10.2.1 begin()	39
5.10.2.2 beginDaylight()	40
5.10.2.3 beginInfrared()	40
5.10.2.4 getAverageValue()	41
5.10.2.5 getBrightest()	42
5.10.2.6 getValue()	42
5.10.2.7 readDLPT()	43
5.10.2.8 readIRPT()	44
5.10.3 Member Data Documentation	44
5.10.3.1 DL_PT_BOTTOM_ADC	44
5.10.3.2 DL_PT_ENABLE	44
5.10.3.3 DL_PT_FRONT_ADC	44
5.10.3.4 IR_PT_BACK_ADC	44
5.10.3.5 IR_PT_ENABLE	45
5.10.3.6 IR_PT_FRONT_ADC	45
5.10.3.7 IR_PT_LEFT_ADC	45
5.10.3.8 IR_PT_RIGHT_ADC	45
5.11 Motion Class Reference	46
5.11.1 Detailed Description	47
5.11.2 Member Function Documentation	47
5.11.2.1 begin()	47
5.11.2.2 leftMotorTask()	48
5.11.2.3 move()	48
5.11.2.4 moveTask()	49
5.11.2.5 moveWithoutCorrection()	50
5.11.2.6 rightMotorTask()	50
5.11.2.7 rotateAntiClockwise()	50
5.11.2.8 rotateClockwise()	51
5.11.2.9 stop()	52
5.11.3 Member Data Documentation	52
5.11.3.1 buffer	52
5.11.3.2 correctionThreshold	52
5.11.3.3 detection	52
5.11.3.4 left	53
5.11.3.5 LEFT_MOTOR_DUTY	53
5.11.3.6 MOTOR_LEFT_PIN	53

5.11.3.7 MOTOR_RIGHT_PIN . . . . .	53
5.11.3.8 right . . . . .	53
5.11.3.9 RIGHT_MOTOR_DUTY . . . . .	53
5.11.3.10 xAntiClockwiseTaskHandle . . . . .	54
5.11.3.11 xClockwiseTaskHandle . . . . .	54
5.11.3.12 xLastWakeTime . . . . .	54
5.11.3.13 xMoveTaskHandle . . . . .	54
5.12 MotionDetection Class Reference . . . . .	54
5.12.1 Detailed Description . . . . .	56
5.12.2 Member Enumeration Documentation . . . . .	56
5.12.2.1 registerBank . . . . .	56
5.12.3 Constructor & Destructor Documentation . . . . .	56
5.12.3.1 MotionDetection() . . . . .	56
5.12.4 Member Function Documentation . . . . .	56
5.12.4.1 begin() . . . . .	57
5.12.4.2 calibrateZAxis() . . . . .	57
5.12.4.3 cmdRead() [1/2] . . . . .	58
5.12.4.4 cmdRead() [2/2] . . . . .	58
5.12.4.5 cmdWrite() [1/2] . . . . .	58
5.12.4.6 cmdWrite() [2/2] . . . . .	58
5.12.4.7 end() . . . . .	59
5.12.4.8 getAcceleration() . . . . .	59
5.12.4.9 getDataFromFIFO() . . . . .	60
5.12.4.10 getRotation() . . . . .	60
5.12.4.11 getTemperature() . . . . .	61
5.12.4.12 getTilt() . . . . .	61
5.12.4.13 getTiltDirection() . . . . .	62
5.12.4.14 getWhoAmI() . . . . .	63
5.12.4.15 initFIFO() . . . . .	63
5.12.4.16 isShaken() . . . . .	64
5.12.4.17 readDoubleRegister() . . . . .	64
5.12.4.18 readFromRegisterBank() . . . . .	65
5.12.4.19 readRegister() . . . . .	65
5.12.4.20 resetRegisterBankAccess() . . . . .	66
5.12.4.21 writeRegister() . . . . .	66
5.12.4.22 writeToRegisterBank() . . . . .	67
5.12.5 Member Data Documentation . . . . .	67
5.12.5.1 buf . . . . .	67
5.12.5.2 bufferLength . . . . .	67
5.12.5.3 defaultShakeThreshold . . . . .	68
5.12.5.4 frequency . . . . .	68
5.12.5.5 gForceCalib . . . . .	68

---

5.12.5.6 handler	68
5.13 Motor Class Reference	68
5.13.1 Detailed Description	69
5.13.2 Constructor & Destructor Documentation	69
5.13.2.1 Motor()	69
5.13.3 Member Function Documentation	69
5.13.3.1 begin()	69
5.13.3.2 getSpeed()	70
5.13.3.3 setSpeed()	70
5.13.4 Member Data Documentation	71
5.13.4.1 channel	71
5.13.4.2 duty	71
5.13.4.3 pin	72
5.13.4.4 timer	72
5.14 MultiColorLight Class Reference	72
5.14.1 Detailed Description	73
5.14.2 Constructor & Destructor Documentation	73
5.14.2.1 MultiColorLight()	73
5.14.3 Member Function Documentation	73
5.14.3.1 begin()	73
5.14.3.2 blink()	74
5.14.3.3 color()	74
5.14.3.4 setLed() [1/3]	75
5.14.3.5 setLed() [2/3]	76
5.14.3.6 setLed() [3/3]	76
5.14.3.7 setTopLeds() [1/2]	77
5.14.3.8 setTopLeds() [2/2]	78
5.14.3.9 turnOffLed()	78
5.14.4 Member Data Documentation	79
5.14.4.1 ledAmount	79
5.14.4.2 ledPin	79
5.14.4.3 maxBrightness	79
5.14.4.4 rgbLeds	80
5.15 Orientation Struct Reference	80
5.15.1 Detailed Description	80
5.15.2 Member Data Documentation	80
5.15.2.1 xRotation	80
5.15.2.2 yRotation	80
5.16 VEML_CONFIG Struct Reference	81
5.16.1 Detailed Description	81
5.16.2 Member Data Documentation	81
5.16.2.1 enabled	81

---



5.16.2.2 exposureTime . . . . .	81
5.16.2.3 mode . . . . .	81
<b>6 File Documentation</b>	<b>83</b>
6.1 doxymain.md File Reference . . . . .	84
6.2 example/advanced/Ampel1/Ampel1.ino File Reference . . . . .	84
6.3 example/advanced/Ampel2/Ampel2.ino File Reference . . . . .	84
6.4 example/advanced/Ampel3/Ampel3.ino File Reference . . . . .	84
6.5 example/advanced/FindAFriend/FindAFriend.ino File Reference . . . . .	84
6.6 example/advanced/FrequencyFindAFriend/FrequencyFindAFriend/FrequencyFindAFriend.ino File Reference . . . . .	84
6.7 example/advanced/phototaxis/phototaxis.ino File Reference . . . . .	84
6.8 example/advanced/simpleMorse/simpleMorse.ino File Reference . . . . .	84
6.9 example/advanced/wuerfeln/wuerfeln.ino File Reference . . . . .	84
6.10 example/advanced/zaehlen/zaehlen.ino File Reference . . . . .	84
6.11 example/color_detection/color_detection.ino File Reference . . . . .	84
6.12 example/display/basic/basic/basic.ino File Reference . . . . .	84
6.13 example/Fernbedienung/empfaenger/empfaenger.ino File Reference . . . . .	84
6.14 example/Fernbedienung/sender/sender.ino File Reference . . . . .	84
6.15 example/IMU/Back_to_Origin/Back_to_Origin.ino File Reference . . . . .	84
6.16 example/IMU/Detection_Print/Detection_Print.ino File Reference . . . . .	84
6.17 example/IMU/Motion_Correction/motion_correction/motion_correction.ino File Reference . . . . .	84
6.18 example/IMU/Shake_Detection/shake_detection/shake_detection.ino File Reference . . . . .	84
6.19 example/IMU/Tilt_Detection/tilt_detection/tilt_detection.ino File Reference . . . . .	84
6.20 example/IMU/Upside_Downside_Detection/Upside_Downside_Detection.ino File Reference . . . . .	84
6.21 example/IRDirection/IRDirection/IRDirection.ino File Reference . . . . .	84
6.22 example/Led/ColorCycle/ColorCycle/ColorCycle.ino File Reference . . . . .	84
6.23 example/lightdetection_serial/lightdetection_serial.ino File Reference . . . . .	84
6.24 example/motion_indicating/motion_indicating.ino File Reference . . . . .	84
6.25 example/motion_minimum/motion_minimum.ino File Reference . . . . .	84
6.26 example/start/start.ino File Reference . . . . .	84
6.27 README.md File Reference . . . . .	84
6.28 src/colorDetection/ColorDetection.cpp File Reference . . . . .	84
6.29 src/colorDetection/ColorDetection.h File Reference . . . . .	85
6.29.1 Enumeration Type Documentation . . . . .	86
6.29.1.1 color . . . . .	86
6.29.1.2 duration . . . . .	87
6.29.1.3 vemlMode . . . . .	87
6.30 src/communication/Communication.cpp File Reference . . . . .	87
6.30.1 Function Documentation . . . . .	88
6.30.1.1 changedConnectionCallback() . . . . .	88
6.30.1.2 newConnectionCallback() . . . . .	89
6.30.1.3 nodeTimeAdjustedCallback() . . . . .	89

---

6.30.1.4 vTaskUpdate()	89
6.30.2 Variable Documentation	90
6.30.2.1 mesh	90
6.30.2.2 userScheduler	90
6.31 src/communication/Communication.h File Reference	90
6.31.1 Macro Definition Documentation	91
6.31.1.1 MESH_PASSWORD	91
6.31.1.2 MESH_PORT	91
6.31.1.3 MESH_PREFIX	92
6.32 src/Dezibot.cpp File Reference	92
6.32.1 Macro Definition Documentation	92
6.32.1.1 SCL_PIN	92
6.32.1.2 SDA_PIN	92
6.33 src/Dezibot.h File Reference	93
6.33.1 Detailed Description	93
6.34 src/display/CharTable.h File Reference	94
6.34.1 Detailed Description	94
6.34.2 Variable Documentation	94
6.34.2.1 font8x8_colwise	95
6.35 src/display/Display.cpp File Reference	95
6.35.1 Detailed Description	95
6.36 src/display/Display.h File Reference	96
6.37 src/display/DisplayCMDs.h File Reference	97
6.37.1 Macro Definition Documentation	98
6.37.1.1 activateDisplay	98
6.37.1.2 addressingMode	98
6.37.1.3 cmd_byte	98
6.37.1.4 colRange	98
6.37.1.5 completeOn	98
6.37.1.6 data_byte	99
6.37.1.7 disableDisplay	99
6.37.1.8 muxRatio	99
6.37.1.9 pageRange	99
6.37.1.10 setChargePump	99
6.37.1.11 setComDirectionFlipped	99
6.37.1.12 setComDirectionNormal	100
6.37.1.13 setComHardwareConfig	100
6.37.1.14 setContrast	100
6.37.1.15 setInverseMode	100
6.37.1.16 setNormalMode	100
6.37.1.17 setOffset	100
6.37.1.18 setOscFreq	101

---

6.37.1.19 setSegmentMap	101
6.37.1.20 setSegmentReMap	101
6.37.1.21 setStartLine	101
6.37.1.22 stopCompleteOn	101
6.38 src/infraredLight/InfraredLED.cpp File Reference	102
6.38.1 Macro Definition Documentation	102
6.38.1.1 pwmSpeedMode	102
6.39 src/infraredLight/InfraredLight.cpp File Reference	103
6.40 src/infraredLight/InfraredLight.h File Reference	103
6.40.1 Detailed Description	104
6.41 src/lightDetection/LightDetection.cpp File Reference	105
6.42 src/lightDetection/LightDetection.h File Reference	105
6.42.1 Detailed Description	106
6.42.2 Enumeration Type Documentation	107
6.42.2.1 photoTransistors	107
6.42.2.2 ptType	107
6.43 src/motion/Motion.cpp File Reference	107
6.43.1 Detailed Description	108
6.44 src/motion/Motion.h File Reference	108
6.44.1 Detailed Description	109
6.44.2 Macro Definition Documentation	110
6.44.2.1 CHANNEL_LEFT	110
6.44.2.2 CHANNEL_RIGHT	110
6.44.2.3 DEFAULT_BASE_VALUE	110
6.44.2.4 DUTY_RES	110
6.44.2.5 FREQUENCY	111
6.44.2.6 LEDC_MODE	111
6.44.2.7 TIMER	111
6.45 src/motion/Motor.cpp File Reference	111
6.46 src/motionDetection/IMU_CMDs.h File Reference	112
6.46.1 Macro Definition Documentation	113
6.46.1.1 ACCEL_DATA_X_HIGH	113
6.46.1.2 ACCEL_DATA_X_LOW	113
6.46.1.3 ACCEL_DATA_Y_HIGH	113
6.46.1.4 ACCEL_DATA_Y_LOW	113
6.46.1.5 ACCEL_DATA_Z_HIGH	114
6.46.1.6 ACCEL_DATA_Z_LOW	114
6.46.1.7 ADDR_MASK	114
6.46.1.8 BLK_SEL_R	114
6.46.1.9 BLK_SEL_W	114
6.46.1.10 CMD_READ	114
6.46.1.11 CMD_WRITE	115

6.46.1.12 FIFO_CONFIG1	115
6.46.1.13 FIFO_CONFIG2	115
6.46.1.14 FIFO_CONFIG5	115
6.46.1.15 FIFO_COUNTH	115
6.46.1.16 FIFO_COUNTL	115
6.46.1.17 FIFO_DATA	116
6.46.1.18 GYRO_DATA_X_HIGH	116
6.46.1.19 GYRO_DATA_X_LOW	116
6.46.1.20 GYRO_DATA_Y_HIGH	116
6.46.1.21 GYRO_DATA_Y_LOW	116
6.46.1.22 GYRO_DATA_Z_HIGH	116
6.46.1.23 GYRO_DATA_Z_LOW	117
6.46.1.24 INTF_CONFIG0	117
6.46.1.25 M_R	117
6.46.1.26 M_W	117
6.46.1.27 MADDR_R	117
6.46.1.28 MADDR_W	117
6.46.1.29 MCLK_RDY	118
6.46.1.30 PWR_MGMT0	118
6.46.1.31 REG_TEMP_HIGH	118
6.46.1.32 REG_TEMP_LOW	118
6.46.1.33 TMST_CONFIG1	118
6.46.1.34 WHO_AM_I	118
6.47 src/motionDetection/MotionDetection.cpp File Reference	119
6.48 src/motionDetection/MotionDetection.h File Reference	119
6.48.1 Detailed Description	120
6.48.2 Enumeration Type Documentation	121
6.48.2.1 Axis	121
6.48.2.2 Direction	121
6.49 src/multiColorLight/ColorConstants.h File Reference	122
6.50 src/multiColorLight/MultiColorLight.cpp File Reference	122
6.51 src/multiColorLight/MultiColorLight.h File Reference	123
6.51.1 Detailed Description	124
6.51.2 Enumeration Type Documentation	124
6.51.2.1 leds	124

# 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 arduinolibary 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 separate folder under src/ that is named equivalent 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 design, 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 contains 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:

<a href="#">averageMeasurement</a>	11
<a href="#">ColorDetection</a>	12
<a href="#">Communication</a>	15
<a href="#">Dezibot</a>	17
<a href="#">Display</a>	20
<a href="#">FIFO_Package</a>	29
<a href="#">IMUResult</a>	30
<a href="#">InfraredLED</a>	31
<a href="#">InfraredLight</a>	36
<a href="#">LightDetection</a>	38
<a href="#">Motion</a>	46
<a href="#">MotionDetection</a>	54
<a href="#">Motor</a>	68
<a href="#">MultiColorLight</a>	72
<a href="#">Orientation</a>	80
<a href="#">VEML_CONFIG</a>	81



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

example/advanced/Ampel1/ <a href="#">Ampel1.ino</a>	84
example/advanced/Ampel2/ <a href="#">Ampel2.ino</a>	84
example/advanced/Ampel3/ <a href="#">Ampel3.ino</a>	84
example/advanced/FindAFriend/ <a href="#">FindAFriend.ino</a>	84
example/advanced/FrequencyFindAFriend/ <a href="#">FrequencyFindAFriend.ino</a>	84
example/advanced/phototaxis/ <a href="#">phototaxis.ino</a>	84
example/advanced/simpleMorse/ <a href="#">simpleMorse.ino</a>	84
example/advanced/wuerfeln/ <a href="#">wuerfeln.ino</a>	84
example/advanced/zaehlen/ <a href="#">zaehlen.ino</a>	84
example/color_detection/ <a href="#">color_detection.ino</a>	84
example/display/basic/basic/ <a href="#">basic.ino</a>	84
example/Fernbedienung/empfaenger/ <a href="#">empfaenger.ino</a>	84
example/Fernbedienung/sender/ <a href="#">sender.ino</a>	84
example/IMU/Back_to_Origin/ <a href="#">Back_to_Origin.ino</a>	84
example/IMU/Detection_Print/ <a href="#">Detection_Print.ino</a>	84
example/IMU/Motion_Correction/motion_correction/ <a href="#">motion_correction.ino</a>	84
example/IMU/Shake_Detection/shake_detection/ <a href="#">shake_detection.ino</a>	84
example/IMU/Tilt_Detection/tilt_detection/ <a href="#">tilt_detection.ino</a>	84
example/IMU/Upside_Downside_Detection/ <a href="#">Upside_Downside_Detection.ino</a>	84
example/IRDirection/IRDirection/ <a href="#">IRDirection.ino</a>	84
example/Led/ColorCycle/ColorCycle/ <a href="#">ColorCycle.ino</a>	84
example/lightdetection_serial/ <a href="#">lightdetection_serial.ino</a>	84
example/motion_indicating/ <a href="#">motion_indicating.ino</a>	84
example/motion_minimum/ <a href="#">motion_minimum.ino</a>	84
example/start/ <a href="#">start.ino</a>	84
src/ <a href="#">Dezibot.cpp</a>	92
src/ <a href="#">Dezibot.h</a>	93
src/colorDetection/ <a href="#">ColorDetection.cpp</a>	84
src/colorDetection/ <a href="#">ColorDetection.h</a>	85
src/communication/ <a href="#">Communication.cpp</a>	87
src/communication/ <a href="#">Communication.h</a>	90
src/display/ <a href="#">CharTable.h</a>	
LookUpTable for 8x8 Pixel Characters for an SSD1306 <a href="#">Display</a>	94
src/display/ <a href="#">Display.cpp</a>	
Adds the ability to print to the display of the robot	95

src/display/Display.h	96
src/display/DisplayCMDs.h	97
src/infraredLight/InfraredLED.cpp	102
src/infraredLight/InfraredLight.cpp	103
src/infraredLight/InfraredLight.h	
Adds the ability to print to the display of the robot	103
src/lightDetection/LightDetection.cpp	105
src/lightDetection/LightDetection.h	
Class for Reading the values of the different Phototransistors, both IR, and DaylightSensors are supported	105
src/motion/Motion.cpp	
Implementation of the <a href="#">Motion</a> class	107
src/motion/Motion.h	
This component controls the ability to rotate and change position	108
src/motion/Motor.cpp	111
src/motionDetection/IMU_CMDs.h	112
src/motionDetection/MotionDetection.cpp	119
src/motionDetection/MotionDetection.h	
This component controls the IMU (Accelerometer & Gyroscope) ICM-42670-P	119
src/multiColorLight/ColorConstants.h	122
src/multiColorLight/MultiColorLight.cpp	122
src/multiColorLight/MultiColorLight.h	
This component controls the ability to show multicolored light, using the RGB-LEDs	123

## 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 RGBW 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

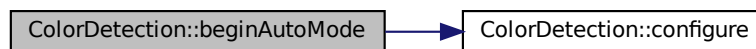
#### 5.2.2.1 beginAutoMode()

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

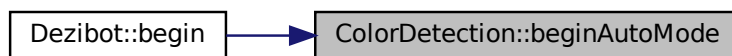
Start RGBW sensor with default configuration.

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 )
```

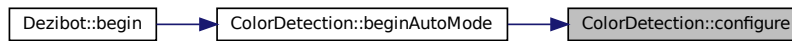
Begin RGBW sensor with passed configuration values.

**Parameters**

<i>config</i>	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**

<i>color</i>	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 Component, 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

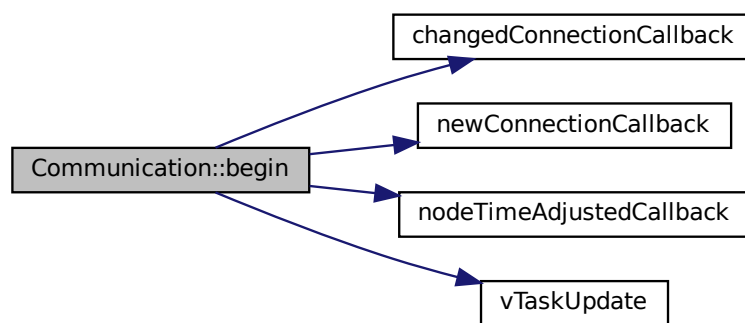
### 5.3.2.1 begin()

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

initialize the Mesh Component, 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.

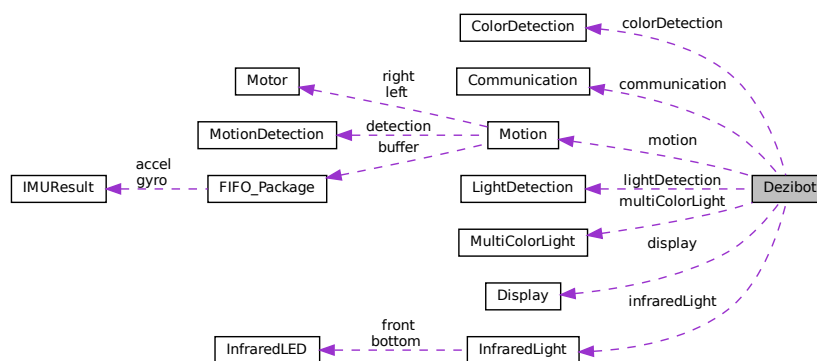
The documentation for this class was generated from the following files:

- 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.

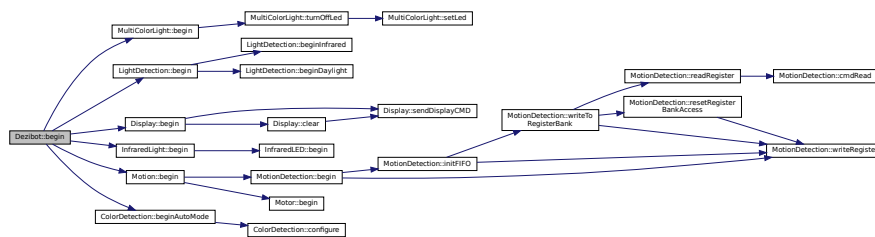
## 5.4.3 Member Function Documentation

### 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 sends 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 )
```

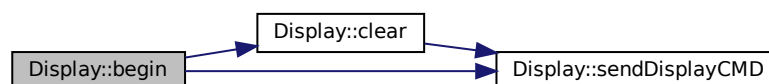
initializes the display datastructures and sends the required cmds to start the display. Should only be called once.

#### Warning

doesn't initialize the I<sup>2</sup>C 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 )
```

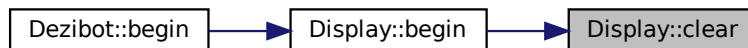
delets all content from the display, resets the linecounter, new print will start at the top left. Orientationflip is not resetted

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

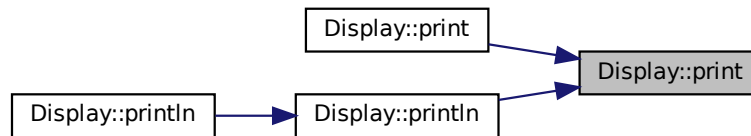
<i>value</i>	the string "xyz" that should be printed to the display
--------------	--

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 )
```

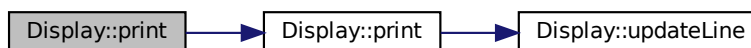
prints the passed string right behind the current displaycontent the sequence "\n" can be used to make a linebreak on the display

#### Parameters

<i>value</i>	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 )
```

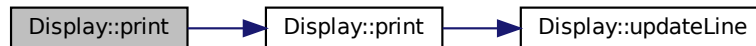
prints the passed string right behind the current displaycontent the sequence "\n" can be used to make a linebreak on the display

**Parameters**

<i>value</i>	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 )
```

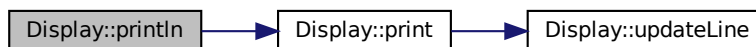
same as the print method, but after the string a line break is inserted

**Parameters**

<i>value</i>	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

<i>value</i>	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

<i>value</i>	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]
```

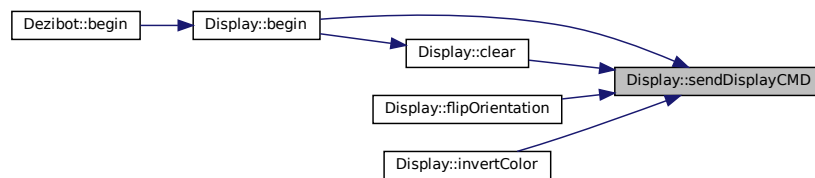
sends the passed cmd to the display, cmd\_byte is added as prefix by the function

#### Parameters

<i>cmd</i>	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

<i>value</i>	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]
```

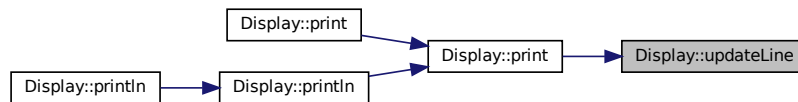
should be called whenever characters where printed to the display. Updates the data of the class to handle line-breaks correctly

**Parameters**

<i>charAmount</i>	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.

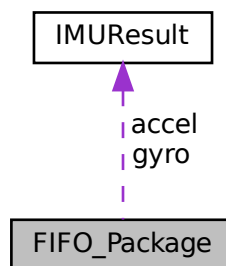
The documentation for this class was generated from the following files:

- [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.

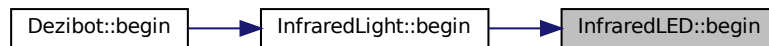
### 5.8.3 Member Function Documentation

### 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

<i>frequency</i>	
------------------	--

Definition at line 53 of file InfraredLED.cpp.

### 5.8.3.3 setState()

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

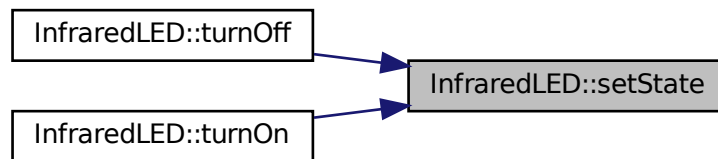
changes state of selected LED depending on the state

#### Parameters

<i>led</i>	which led will be affected
<i>state</i>	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

<i>led</i>	
------------	--

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.

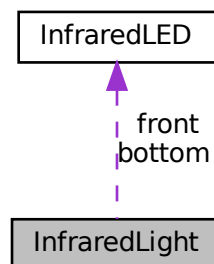
The documentation for this class was generated from the following files:

- [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 distinguish 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

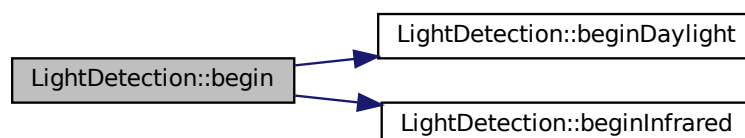
#### 5.10.2.1 begin()

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

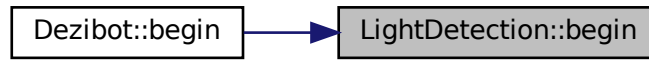
initialize the Lightdetection Component, must be called before the other methods are used.

Definition at line 4 of file LightDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

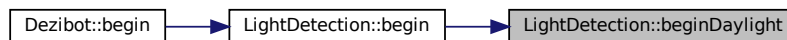


### 5.10.2.2 beginDaylight()

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

Definition at line 75 of file `LightDetection.cpp`.

Here is the caller graph for this function:

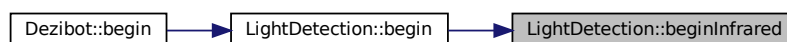


### 5.10.2.3 beginInfrared()

```
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

<i>sensor</i>	Which Phototransistor should be read
<i>measurments</i>	how many measurements should be taken
<i>timeBetween</i>	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 distinguish between IR and Daylight

## Parameters

<i>type</i>	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

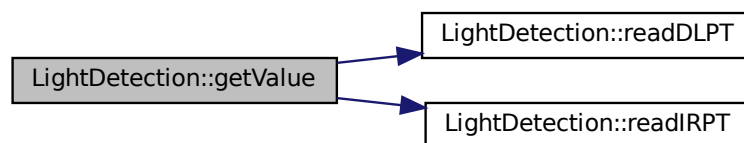
<i>sensor</i>	which sensor to read
---------------	----------------------

## Returns

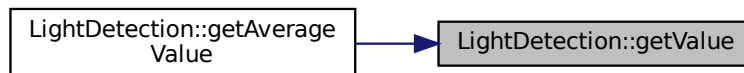
uint the reading of the sensor. between 0-4095

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:

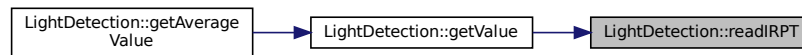


### 5.10.2.8 readIRPT()

```
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.

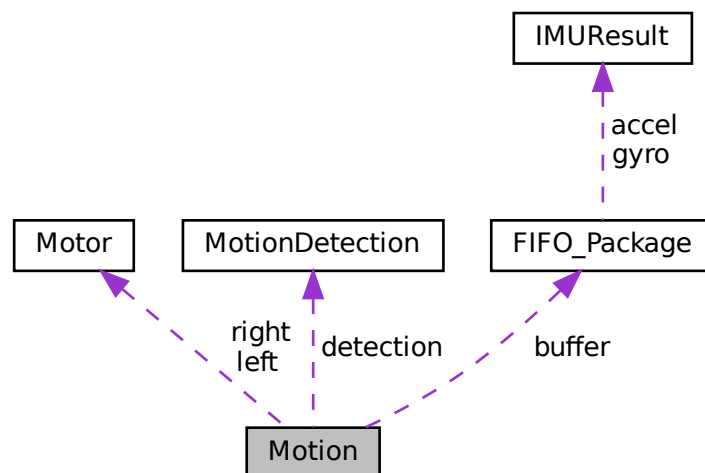
The documentation for this class was generated from the following files:

- [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

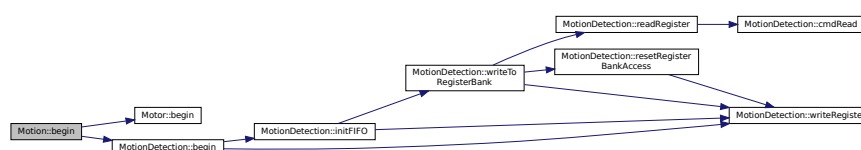
#### 5.11.2.1 begin()

```
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:



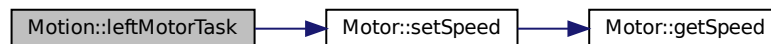
### 5.11.2.2 leftMotorTask()

```

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]
  
```

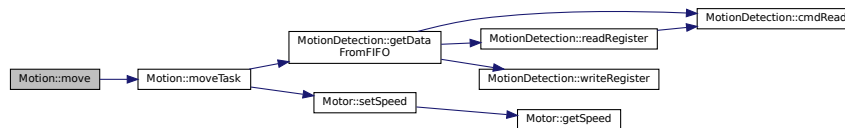
Move forward for a certain amount of time. Call with `moveForMs` 0 will start movement, that must be stopped explicit by call to `stop()`. The function applies a basic algorithm to improve the straightness of the movement. Lifting the robot from the desk may corrupt the results and is not recommended.

Parameters

<i>moveForMs</i>	Representing the duration of forward moving in milliseconds.
<i>baseValue</i>	The value that is used to start with the calibrated movement. Defaults to 3900. If the <a href="#">Dezibot</a> 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:



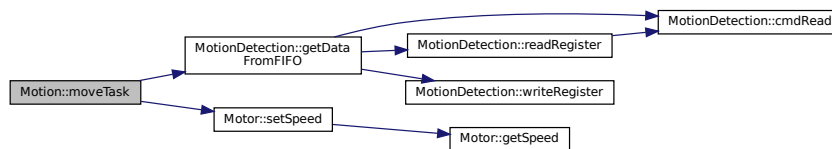
5.11.2.4 moveTask()

```

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

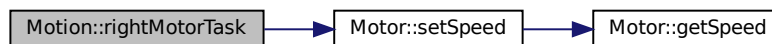
<i>moveForMs</i>	how many ms should the robot move, or 0 to let the robot move until another move command is mentioned, default is 0
<i>baseValue</i>	the duty value that is used for the movement, default is 0

### 5.11.2.6 rightMotorTask()

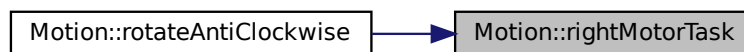
```
void Motion::rightMotorTask (
    void * args ) [static], [protected]
```

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]
```

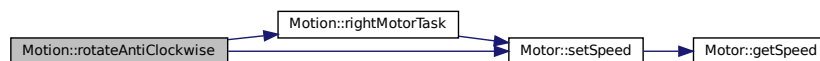
Rotate anticlockwise for a certain amount of time. Call with moveForMs 0 will start movement, that must be stopped explicit by call to [stop\(\)](#).

## Parameters

<i>rotateForMs</i>	Representing the duration of rotating anticlockwise in milliseconds or 0 to let the robot turn until another movecommand is issued. Default is 0.
<i>baseValue</i>	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]
  
```

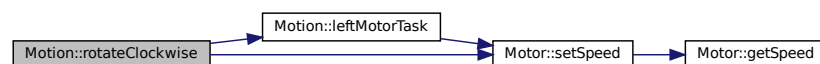
Rotate clockwise for a certain amount of time. Call with `moveForMs` 0 will start movement, that must be stopped explicit by call to `stop()`.

## Parameters

<i>rotateForMs</i>	Representing the duration of rotating clockwise in milliseconds, or 0 to rotate until another movecmd is issued. Default is 0
<i>baseValue</i>	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:



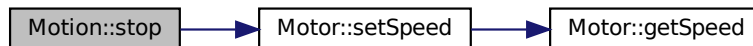
### 5.11.2.9 stop()

```
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 [getWhoAml](#) (void)
 

*Returns the value of reading the whoAml 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 available 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

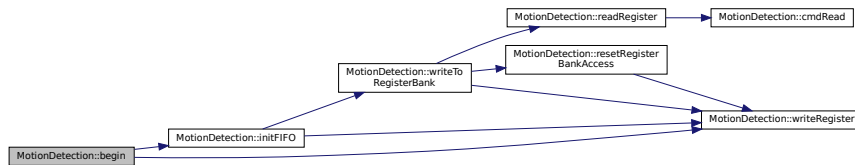
### 5.12.4.1 begin()

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

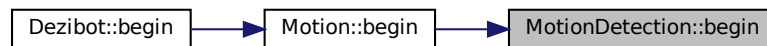
initialized the IMU Component. Wakes the IMU from Standby Set configuration

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 persistent, so the value is not stored when the program is restarted / the robot is powered off

#### Parameters

<i>gforceValue</i>	the value the IMU returns for the gravitationforce -> to get this value, place the robot on a leveled surface and read the value <a href="#">getAcceleration().z</a>
--------------------	--

Definition at line 74 of file MotionDetection.cpp.

### 5.12.4.3 cmdRead() [1/2]

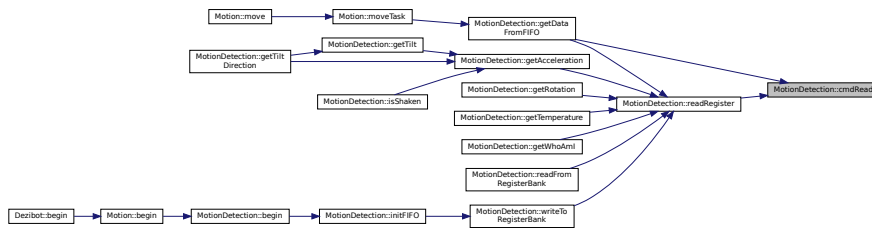
```
uint8_t MotionDetection::cmdRead (
    uint8_t reg ) [protected]
```

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

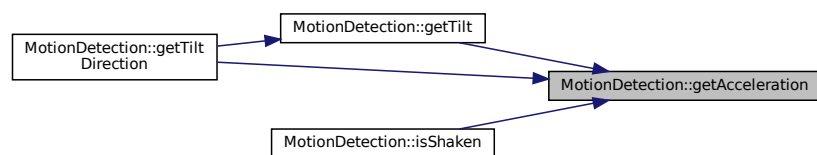
`IMUResult` that contains the new read values

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 available packages from fifo, after 40ms Fifo is full

#### Parameters

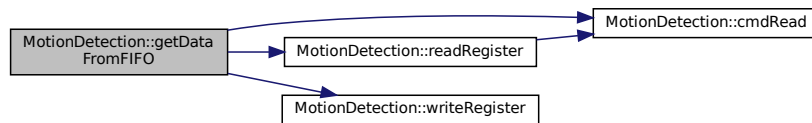
<i>buffer</i>	pointer to <a href="#">FIFO_Package</a> Struct that at least must have size 64 (this is the max package count with APEX Enabled)
---------------	--

#### Returns

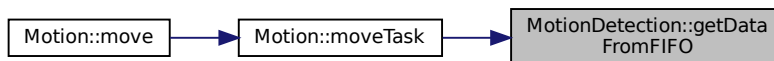
the amount of acutally fetched packages

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.

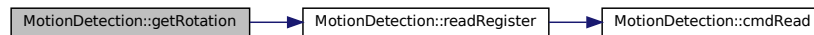


**Returns**

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 )
```

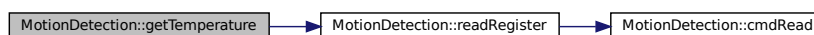
Reads the current On Chip temperature of the IMU.

**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**

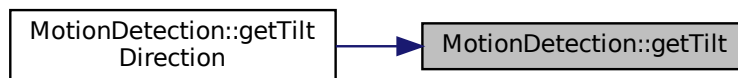
The results are only valid, if the robot is not moved in any way during the measurement, as the calculation is made by using the acceleration values. If it's detected, that the robot is accelerated while measuring, the method will return max(int). Please note that the imu is pretty sensitiv, even walking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

<i>tolerance</i>	(optional, default = 10) how many degrees can the robot be tilted, and still will be considered as neutral.
------------------	---

**Returns**

Direction the direction in that the robot is tilted most. Front is considered 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 )
```

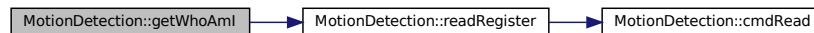
Returns the value of reading the whoAmI register When IMU working correctly, value should be 0x67.

#### 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:

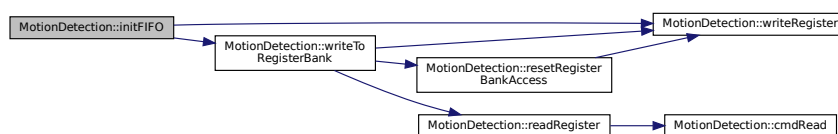


#### 5.12.4.15 initFIFO()

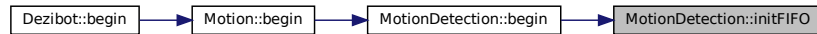
```
void MotionDetection::initFIFO ( ) [protected]
```

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 acceleration values is calculated and checked against threshold. If  $\text{sum} > \text{threshold}$  a shake is detected, else not.

##### Parameters

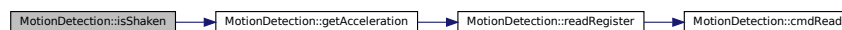
<i>threshold</i>	(optional) the level of acceleration that must be reached to detect a shake
<i>axis</i>	(optional) select which axis should be used for detection. Possible values are <code>xAxis</code> , <code>yAxis</code> , <code>zAxis</code> . It's possible to combine multiple axis with the bitwise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

##### 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()

```

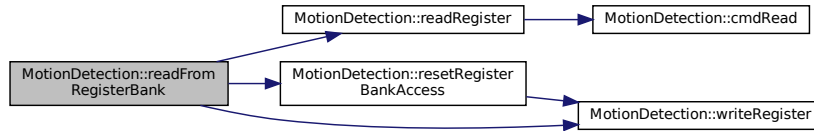
int16_t MotionDetection::readDoubleRegister (
    uint8_t lowerReg ) [protected]
  
```

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()

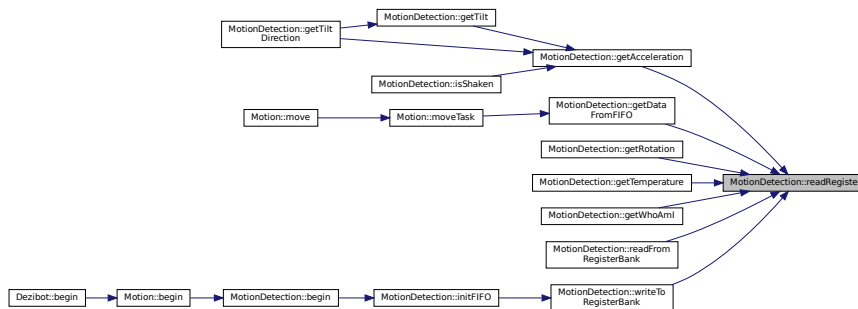
```
uint8_t MotionDetection::readRegister (
    uint8_t reg ) [protected]
```

Definition at line 162 of file MotionDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

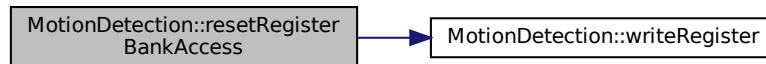


### 5.12.4.20 resetRegisterBankAccess()

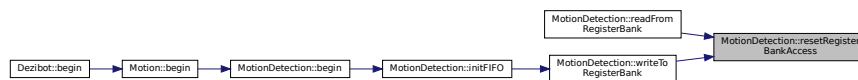
```
void MotionDetection::resetRegisterBankAccess ( ) [protected]
```

Definition at line 224 of file MotionDetection.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

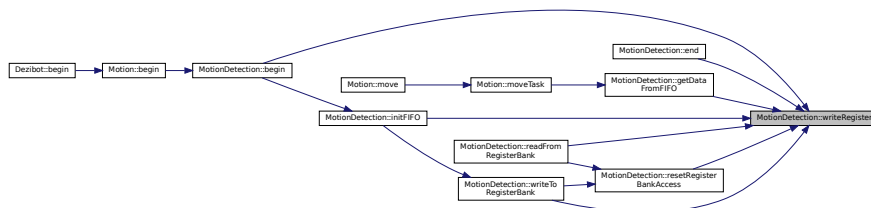


### 5.12.4.21 writeRegister()

```
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:

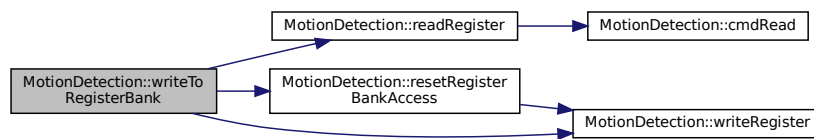


### 5.12.4.22 writeToRegisterBank()

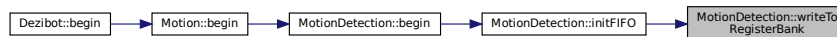
```
void MotionDetection::writeToRegisterBank (
    registerBank bank,
    uint8_t reg,
    uint8_t value ) [protected]
```

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

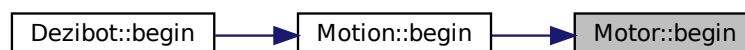
#### 5.13.3.1 begin()

```
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 )
```

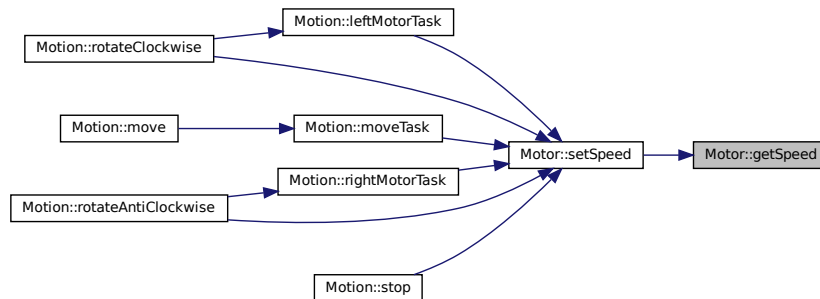
returns the currently activ speed

#### 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 requiried at any time to use that method to access the motors or methods of the motionclass to avoid such peaks.

#### Parameters

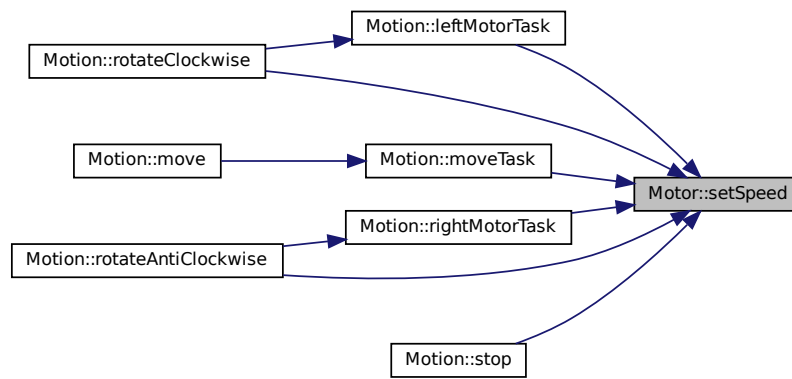
<i>duty</i>	the duty cyle that should be set, can be between 0-8192
-------------	---

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 calculate 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

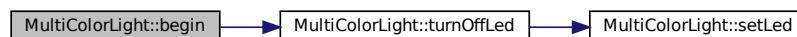
#### 5.14.3.1 begin()

```
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

<i>amount</i>	how often should the leds blink
<i>color</i>	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
<i>leds</i>	which LEDs should blink, default is TOP
<i>interval</i>	how many milliseconds 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 calculate the used colorformat from a rgb-value

#### Parameters

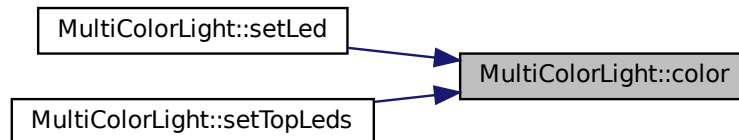
<i>r</i>	red (0-100)
<i>g</i>	green (0-100)
<i>b</i>	blue (0-100)

**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**

<i>leds</i>	which leds should be updated
<i>color</i>	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 )
```

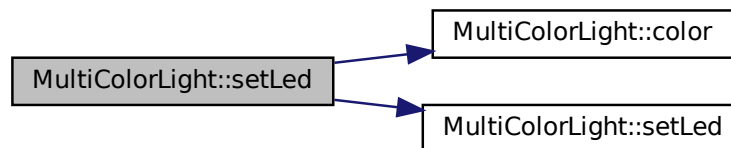
Set the specified leds to the passed color value.

#### Parameters

<i>leds</i>	which leds should be updated
<i>red</i>	brightness of red, is normalized in the function
<i>green</i>	brightness of green, is normalized in the function
<i>blue</i>	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.

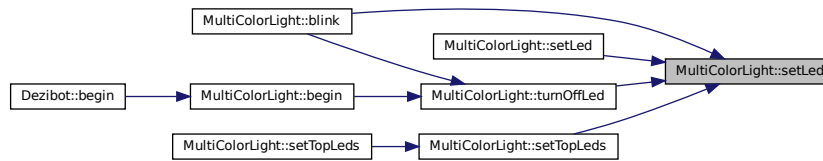
#### Parameters

<i>index</i>	ranging from 0-2, 0: Right, 1: Left, 2: Bottom
<i>color</i>	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

<i>color</i>	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 )
```

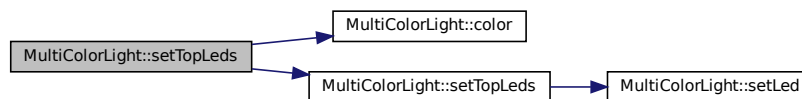
sets the two leds on the top of the robot to the specified color

#### Parameters

<i>red</i>	brightness of red, is normalized in the function
<i>green</i>	brightness of green, is normalized in the function
<i>blue</i>	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

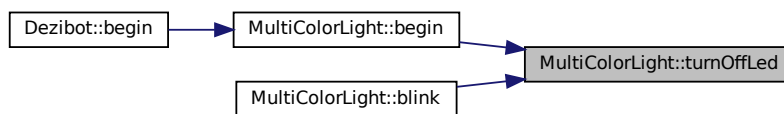
<i>leds</i>	which leds should be turned off, defaults to ALL
-------------	--

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](#)



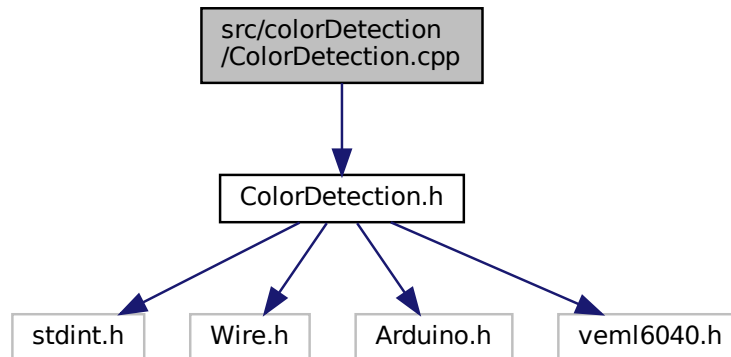
## Chapter 6

# File Documentation

- 6.1 [doxymain.md File Reference](#)
- 6.2 [example/advanced/Ampel1/Ampel1.ino File Reference](#)
- 6.3 [example/advanced/Ampel2/Ampel2.ino File Reference](#)
- 6.4 [example/advanced/Ampel3/Ampel3.ino File Reference](#)
- 6.5 [example/advanced/FindAFriend/FindAFriend.ino File Reference](#)
- 6.6 [example/advanced/FrequencyFindAFriend/FrequencyFindAFriend/  
FrequencyFindAFriend.ino File Reference](#)
- 6.7 [example/advanced/phototaxis/phototaxis.ino File Reference](#)
- 6.8 [example/advanced/simpleMorse/simpleMorse.ino File Reference](#)
- 6.9 [example/advanced/wuerfeln/wuerfeln.ino File Reference](#)
- 6.10 [example/advanced/zaehlen/zaehlen.ino File Reference](#)
- 6.11 [example/color\\_detection/color\\_detection.ino File Reference](#)
- 6.12 [example/display/basic/basic/basic.ino File Reference](#)
- 6.13 [example/Fernbedienung/empfaenger/empfaenger.ino File Reference](#)
- 6.14 [example/Fernbedienung/sender/sender.ino File Reference](#)
- 6.15 [example/IMU/Back\\_to\\_Origin/Back\\_to\\_Origin.ino File Reference](#)
- 6.16 [example/IMU/Detection\\_Print/Detection\\_Print.ino File Reference](#)
- 6.17 [example/IMU/Motion\\_Correction/motion\\_correction/  
motion\\_correction.ino File Reference](#)
- 6.18 [example/IMU/Shake\\_Detection/shake\\_detection/  
shake\\_detection.ino File Reference](#)



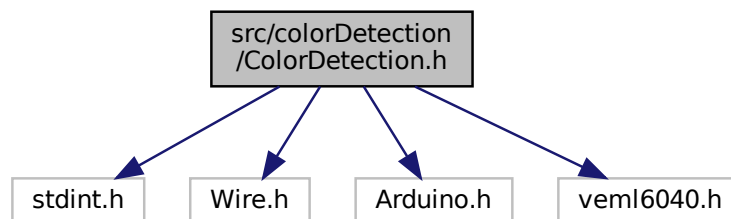
Include dependency graph for ColorDetection.cpp:



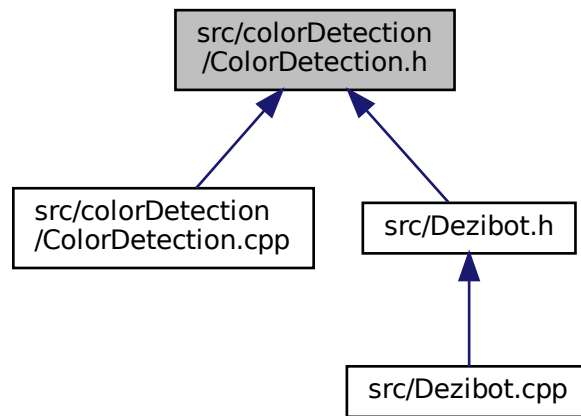
## 6.29 src/colorDetection/ColorDetection.h File Reference

```
#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

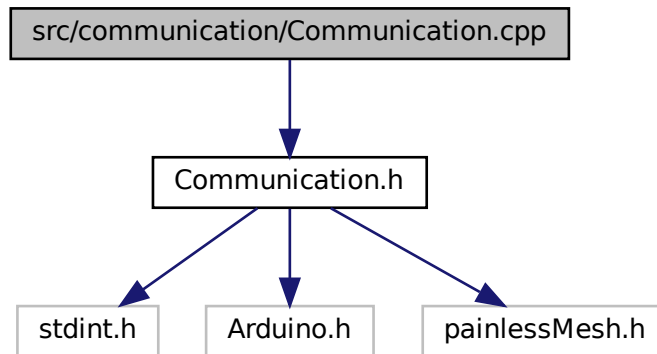
AUTO	
MANUAL	

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

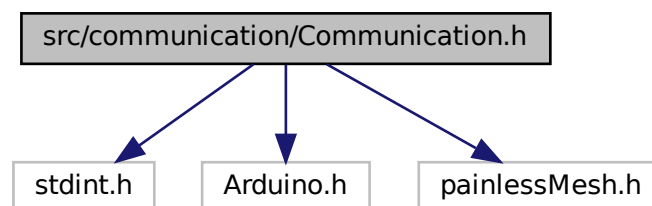
```
Scheduler userScheduler
```

Definition at line 3 of file Communication.cpp.

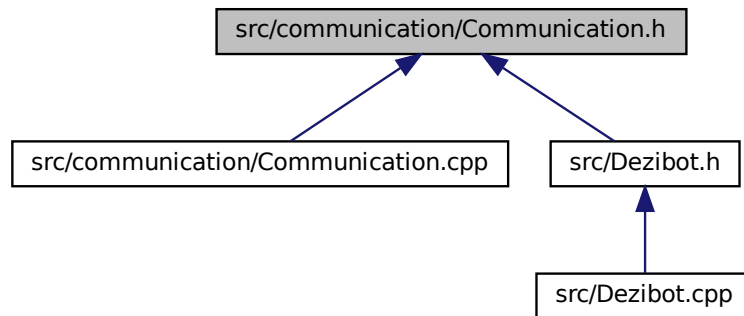
## 6.31 src/communication/Communication.h File Reference

```
#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.

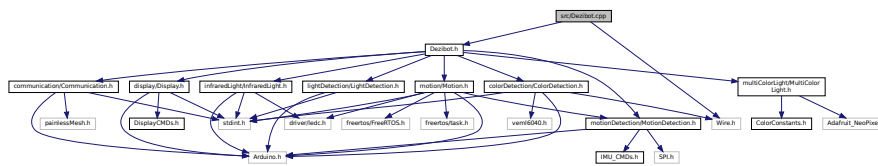
### 6.31.1.3 MESH\_PREFIX

```
#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
```

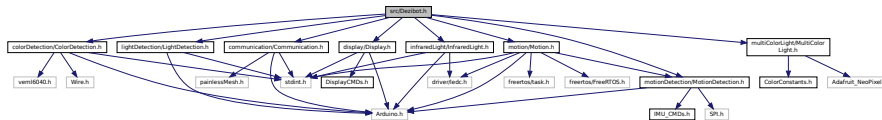
Definition at line 2 of file Dezibot.cpp.



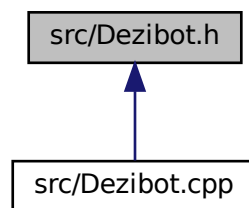
## 6.33 src/Dezibot.h File Reference

```
#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

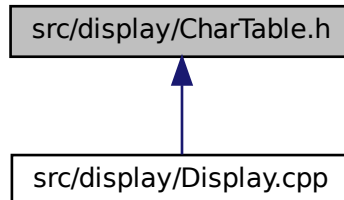
#### Copyright

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](mailto:hans.haupt@dezibot.de) )

#### Version

0.1

#### Date

2024-05-24

#### Copyright

Copyright (c) 2024

### 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.

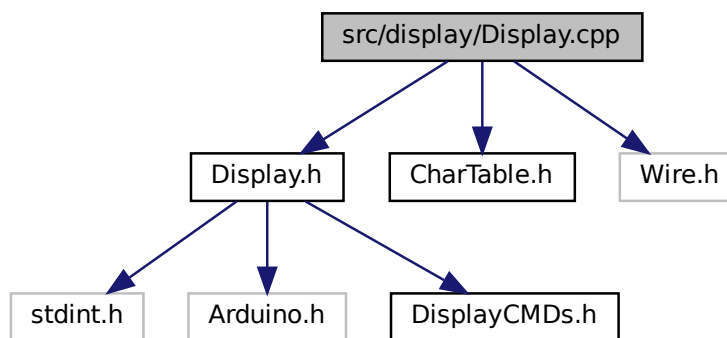
Definition at line 20 of file CharTable.h.

## 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](mailto:hans.haupt@dezibot.de) )

#### Version

0.1

#### Date

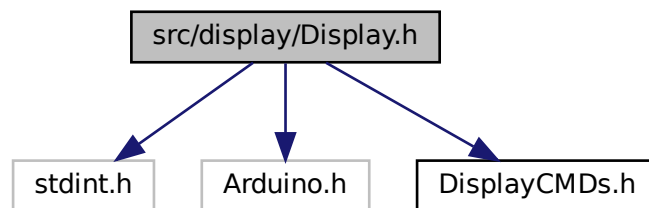
2024-06-05

#### Copyright

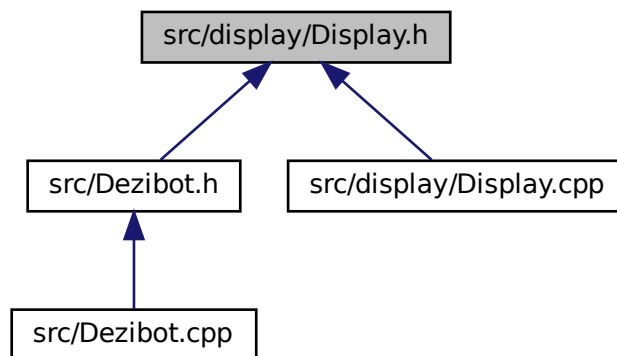
Copyright (c) 2024

## 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:

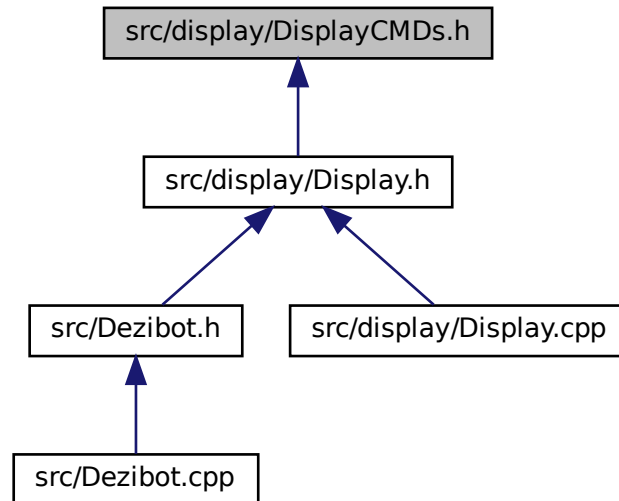


## Classes

- 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

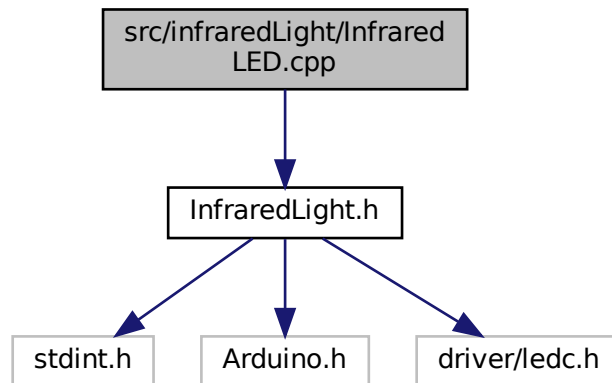
```
#define stopCompleteOn 0xa4
```

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`

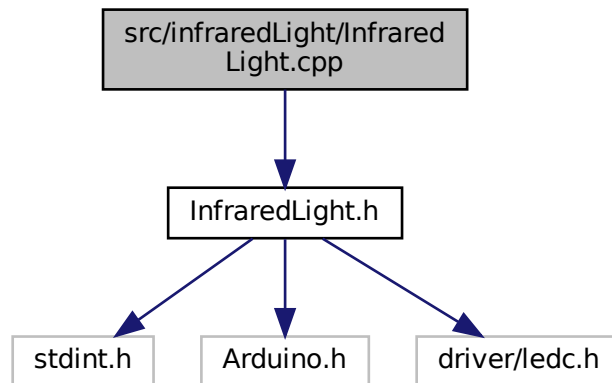
```
#define pwmSpeedMode LEDC_LOW_SPEED_MODE
```

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

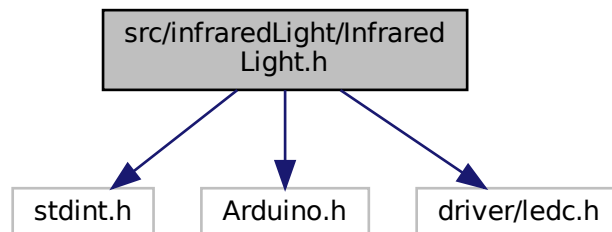
Adds the ability to print to the display of the robot.

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

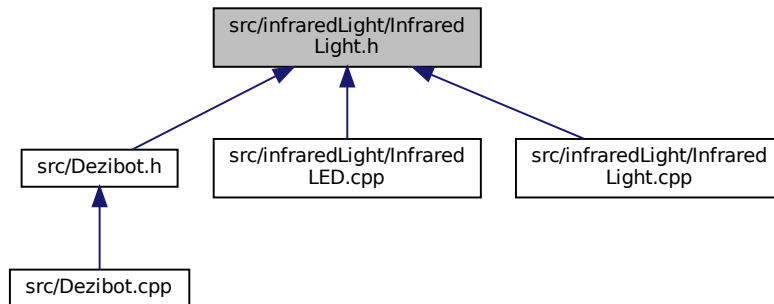
```
#include <Arduino.h>
```

```
#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](mailto:hans.haupt@dezibot.de) )

#### Version

0.1

#### Date

2024-05-24

#### Copyright

Copyright (c) 2024

#### Author

Hans Haupt ( [hans.haupt@dezibot.de](mailto:hans.haupt@dezibot.de) )

#### Version

0.1

#### Date

2024-04-27

#### Copyright

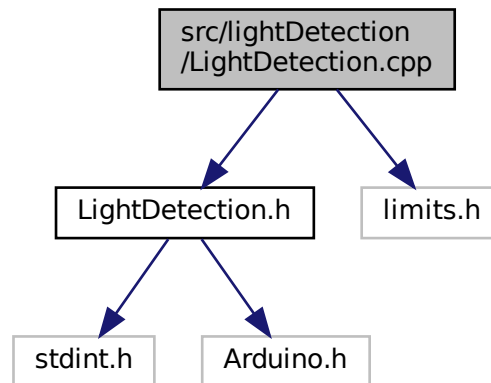
Copyright (c) 2024

## 6.41 src/lightDetection/LightDetection.cpp File Reference

```
#include "LightDetection.h"
```

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

Include dependency graph for LightDetection.cpp:



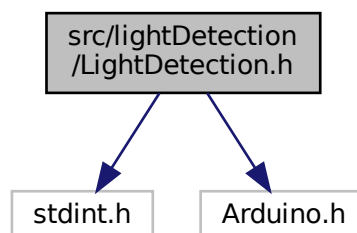
## 6.42 src/lightDetection/LightDetection.h File Reference

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

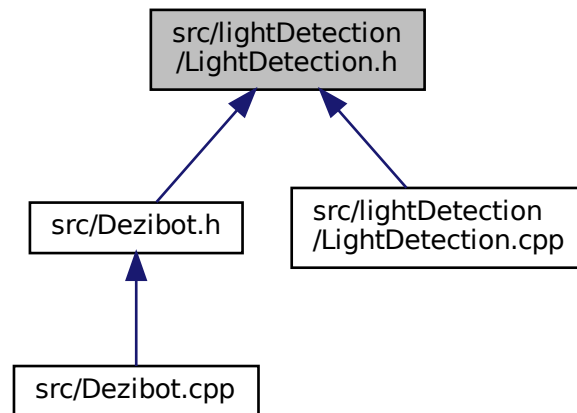
```
#include <stdint.h>
```

```
#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](mailto:hans.haupt@dezibot.de) )

#### Version

0.1

#### Date

2024-04-26

#### Copyright

Copyright (c) 2024

## 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	

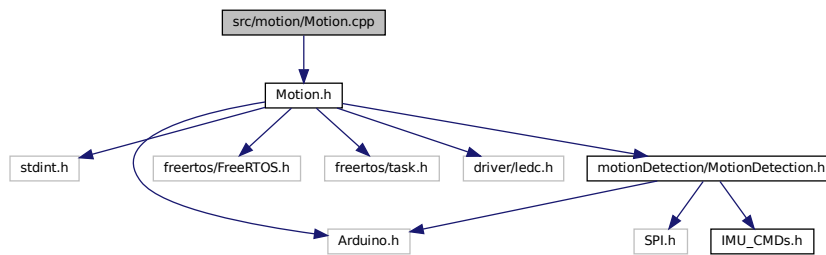
Definition at line 36 of file LightDetection.h.

## 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

Copyright (c) 2023

## 6.44 src/motion/Motion.h File Reference

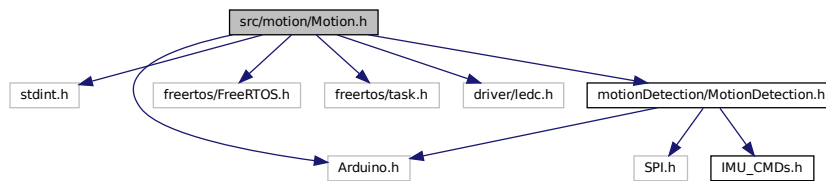
This component controls the ability to rotate and change position.

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

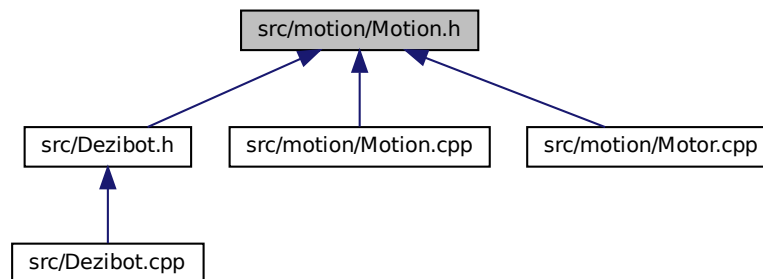


```
#include "motionDetection/MotionDetection.h"
```

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**

Copyright (c) 2023

## 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

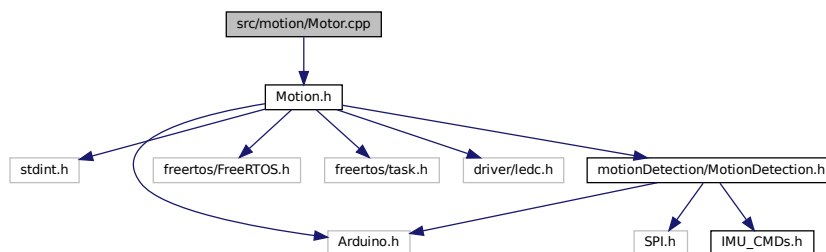
```
#define TIMER LEDC_TIMER_2
```

Definition at line 21 of file Motion.h.

## 6.45 src/motion/Motor.cpp File Reference

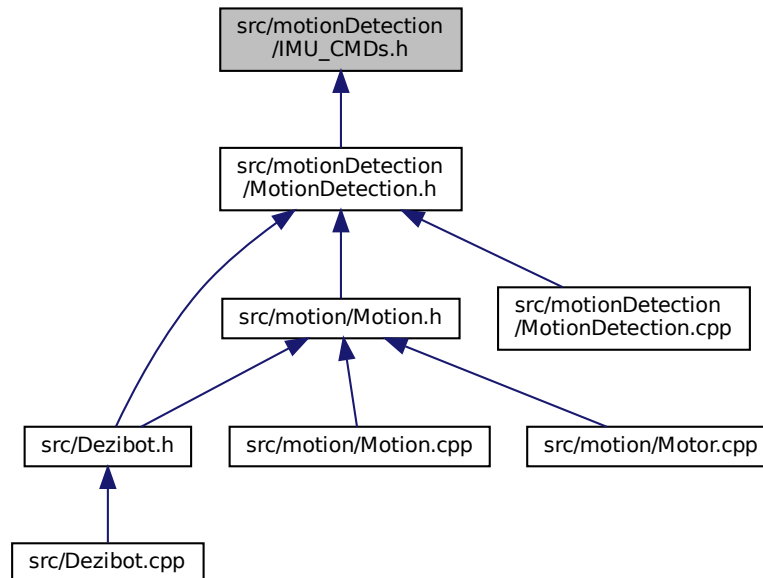
```
#include "Motion.h"
```

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\_COMMANDS.h.

### 6.46.1.2 ACCEL\_DATA\_X\_LOW

```
#define ACCEL_DATA_X_LOW 0x0C
```

Definition at line 15 of file IMU\_COMMANDS.h.

### 6.46.1.3 ACCEL\_DATA\_Y\_HIGH

```
#define ACCEL_DATA_Y_HIGH 0x0D
```

Definition at line 16 of file IMU\_COMMANDS.h.

### 6.46.1.4 ACCEL\_DATA\_Y\_LOW

```
#define ACCEL_DATA_Y_LOW 0x0E
```

Definition at line 17 of file IMU\_COMMANDS.h.

#### 6.46.1.5 ACCEL\_DATA\_Z\_HIGH

```
#define ACCEL_DATA_Z_HIGH 0x0F
```

Definition at line 18 of file IMU\_COMMANDS.h.

#### 6.46.1.6 ACCEL\_DATA\_Z\_LOW

```
#define ACCEL_DATA_Z_LOW 0x10
```

Definition at line 19 of file IMU\_COMMANDS.h.

#### 6.46.1.7 ADDR\_MASK

```
#define ADDR_MASK 0x7F
```

Definition at line 6 of file IMU\_COMMANDS.h.

#### 6.46.1.8 BLK\_SEL\_R

```
#define BLK_SEL_R 0x7C
```

Definition at line 34 of file IMU\_COMMANDS.h.

#### 6.46.1.9 BLK\_SEL\_W

```
#define BLK_SEL_W 0x79
```

Definition at line 33 of file IMU\_COMMANDS.h.

#### 6.46.1.10 CMD\_READ

```
#define CMD_READ 0x80
```

Definition at line 4 of file IMU\_COMMANDS.h.

#### 6.46.1.11 CMD\_WRITE

```
#define CMD_WRITE 0x00
```

Definition at line 5 of file IMU\_COMMANDS.h.

#### 6.46.1.12 FIFO\_CONFIG1

```
#define FIFO_CONFIG1 0x28
```

Definition at line 43 of file IMU\_COMMANDS.h.

#### 6.46.1.13 FIFO\_CONFIG2

```
#define FIFO_CONFIG2 0x29
```

Definition at line 44 of file IMU\_COMMANDS.h.

#### 6.46.1.14 FIFO\_CONFIG5

```
#define FIFO_CONFIG5 0x01
```

Definition at line 47 of file IMU\_COMMANDS.h.

#### 6.46.1.15 FIFO\_COUNTH

```
#define FIFO_COUNTH 0x3D
```

Definition at line 40 of file IMU\_COMMANDS.h.

#### 6.46.1.16 FIFO\_COUNTL

```
#define FIFO_COUNTL 0x3E
```

Definition at line 41 of file IMU\_COMMANDS.h.

#### 6.46.1.17 FIFO\_DATA

```
#define FIFO_DATA 0x3F
```

Definition at line 42 of file IMU\_COMMANDS.h.

#### 6.46.1.18 GYRO\_DATA\_X\_HIGH

```
#define GYRO_DATA_X_HIGH 0x11
```

Definition at line 21 of file IMU\_COMMANDS.h.

#### 6.46.1.19 GYRO\_DATA\_X\_LOW

```
#define GYRO_DATA_X_LOW 0x12
```

Definition at line 22 of file IMU\_COMMANDS.h.

#### 6.46.1.20 GYRO\_DATA\_Y\_HIGH

```
#define GYRO_DATA_Y_HIGH 0x13
```

Definition at line 23 of file IMU\_COMMANDS.h.

#### 6.46.1.21 GYRO\_DATA\_Y\_LOW

```
#define GYRO_DATA_Y_LOW 0x14
```

Definition at line 24 of file IMU\_COMMANDS.h.

#### 6.46.1.22 GYRO\_DATA\_Z\_HIGH

```
#define GYRO_DATA_Z_HIGH 0x15
```

Definition at line 25 of file IMU\_COMMANDS.h.



### 6.46.1.23 GYRO\_DATA\_Z\_LOW

```
#define GYRO_DATA_Z_LOW 0x16
```

Definition at line 26 of file IMU\_COMMANDS.h.

### 6.46.1.24 INTF\_CONFIG0

```
#define INTF_CONFIG0 0x35
```

Definition at line 31 of file IMU\_COMMANDS.h.

### 6.46.1.25 M\_R

```
#define M_R 0x7E
```

Definition at line 38 of file IMU\_COMMANDS.h.

### 6.46.1.26 M\_W

```
#define M_W 0x7B
```

Definition at line 37 of file IMU\_COMMANDS.h.

### 6.46.1.27 MADDR\_R

```
#define MADDR_R 0x7D
```

Definition at line 36 of file IMU\_COMMANDS.h.

### 6.46.1.28 MADDR\_W

```
#define MADDR_W 0x7A
```

Definition at line 35 of file IMU\_COMMANDS.h.

#### 6.46.1.29 MCLK\_RDY

```
#define MCLK_RDY 0x00
```

Definition at line 9 of file IMU\_COMMANDS.h.

#### 6.46.1.30 PWR\_MGMT0

```
#define PWR_MGMT0 0x1F
```

Definition at line 28 of file IMU\_COMMANDS.h.

#### 6.46.1.31 REG\_TEMP\_HIGH

```
#define REG_TEMP_HIGH 0x09
```

Definition at line 12 of file IMU\_COMMANDS.h.

#### 6.46.1.32 REG\_TEMP\_LOW

```
#define REG_TEMP_LOW 0x0A
```

Definition at line 11 of file IMU\_COMMANDS.h.

#### 6.46.1.33 TMST\_CONFIG1

```
#define TMST_CONFIG1 0x00
```

Definition at line 48 of file IMU\_COMMANDS.h.

#### 6.46.1.34 WHO\_AM\_I

```
#define WHO_AM_I 0x75
```

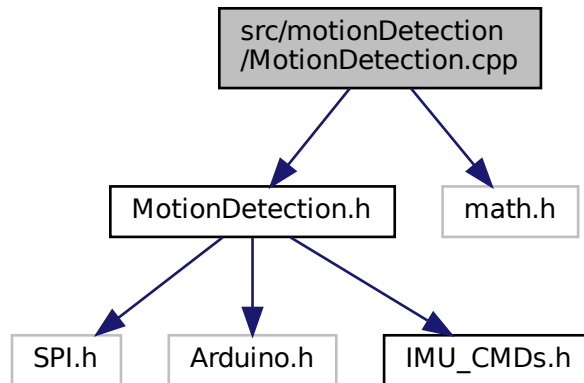
Definition at line 29 of file IMU\_COMMANDS.h.

## 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

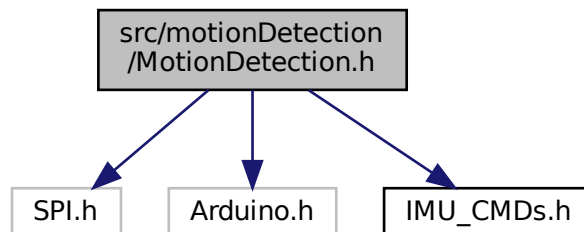
This component controls the IMU (Accelerometer & Gyroscope) ICM-42670-P.

```
#include <SPI.h>
```

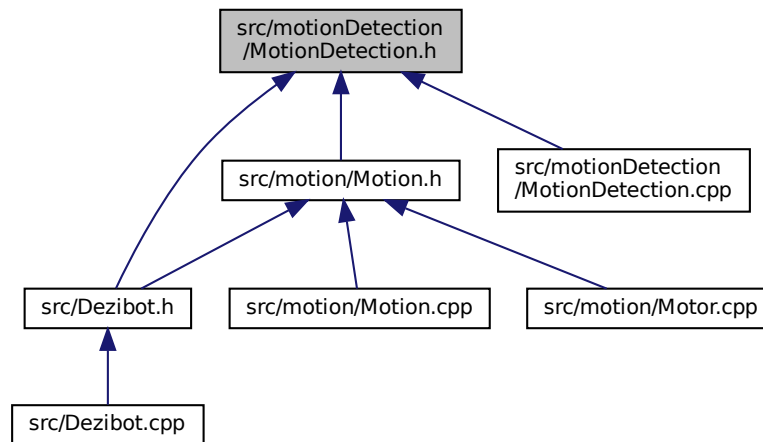
```
#include <Arduino.h>
```

```
#include "IMU_COMMANDS.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

Copyright (c) 2023

## 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](#)

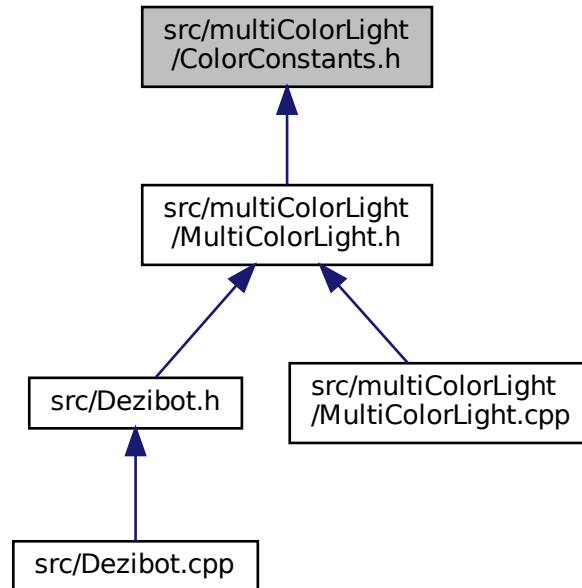
Enumerator

Front	
Left	
Right	
Back	
Neutral	
Flipped	
Error	

Definition at line 32 of file MotionDetection.h.

## 6.49 src/multiColorLight/ColorConstants.h File Reference

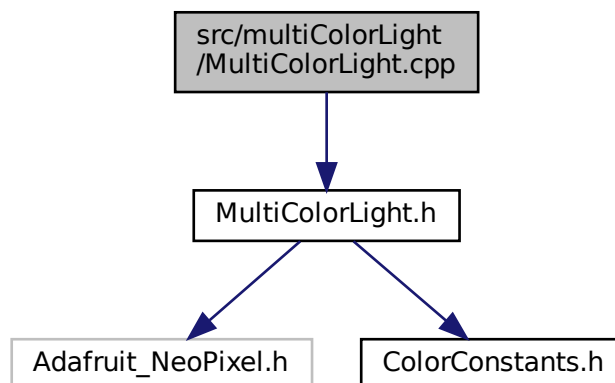
This graph shows which files directly or indirectly include this file:



## 6.50 src/multiColorLight/MultiColorLight.cpp File Reference

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

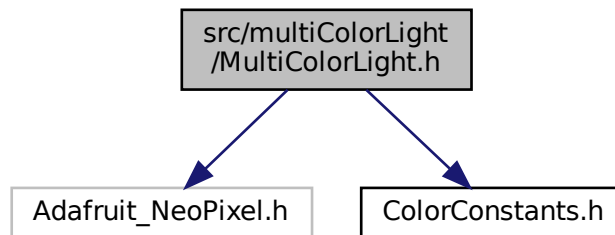
Include dependency graph for `MultiColorLight.cpp`:



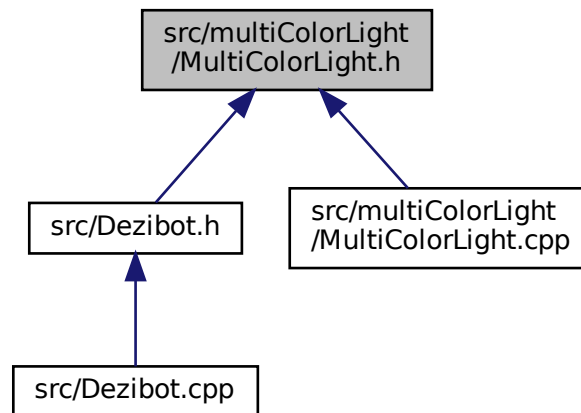
## 6.51 src/multiColorLight/MultiColorLight.h File Reference

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

- accel
  - FIFO\_Package, [29](#)
- ACCEL\_DATA\_X\_HIGH
  - IMU\_COMMANDS.h, [113](#)
- ACCEL\_DATA\_X\_LOW
  - IMU\_COMMANDS.h, [113](#)
- ACCEL\_DATA\_Y\_HIGH
  - IMU\_COMMANDS.h, [113](#)
- ACCEL\_DATA\_Y\_LOW
  - IMU\_COMMANDS.h, [113](#)
- ACCEL\_DATA\_Z\_HIGH
  - IMU\_COMMANDS.h, [113](#)
- ACCEL\_DATA\_Z\_LOW
  - IMU\_COMMANDS.h, [114](#)
- activateDisplay
  - DisplayCOMMANDS.h, [98](#)
- ADDR\_MASK
  - IMU\_COMMANDS.h, [114](#)
- addressingMode
  - DisplayCOMMANDS.h, [98](#)
- ALL
  - MultiColorLight.h, [124](#)
- AUTO
  - ColorDetection.h, [87](#)
- averageMeasurement, [11](#)
  - done, [11](#)
  - measurementAmount, [11](#)
  - result, [12](#)
  - sensor, [12](#)
  - timeBetween, [12](#)
- Axis
  - MotionDetection.h, [121](#)
- Back
  - MotionDetection.h, [121](#)
- begin
  - Communication, [15](#)
  - Dezibot, [18](#)
  - Display, [21](#)
  - InfraredLED, [32](#)
  - InfraredLight, [37](#)
  - LightDetection, [39](#)
  - Motion, [47](#)
  - MotionDetection, [56](#)
  - Motor, [69](#)
  - MultiColorLight, [73](#)
- beginAutoMode
  - ColorDetection, [13](#)
- beginDaylight
  - LightDetection, [40](#)
- beginInfrared
  - LightDetection, [40](#)
- blink
  - MultiColorLight, [73](#)
- BLK\_SEL\_R
  - IMU\_COMMANDS.h, [114](#)
- BLK\_SEL\_W
  - IMU\_COMMANDS.h, [114](#)
- BOTTOM
  - MultiColorLight.h, [124](#)
- bottom
  - InfraredLight, [37](#)
- buf
  - MotionDetection, [67](#)
- buffer
  - Motion, [52](#)
- bufferLength
  - MotionDetection, [67](#)
- calibrateZAxis
  - MotionDetection, [57](#)
- changedConnectionCallback
  - Communication.cpp, [88](#)
- channel
  - InfraredLED, [35](#)
  - Motor, [71](#)
- CHANNEL\_LEFT
  - Motion.h, [110](#)
- CHANNEL\_RIGHT
  - Motion.h, [110](#)
- charsOnCurrLine
  - Display, [28](#)
- CharTable.h
  - font8x8\_colwise, [94](#)
- clear
  - Display, [21](#)
- cmd\_byte
  - DisplayCOMMANDS.h, [98](#)
- CMD\_READ
  - IMU\_COMMANDS.h, [114](#)
- CMD\_WRITE
  - IMU\_COMMANDS.h, [114](#)
- cmdRead
  - MotionDetection, [58](#)
- cmdWrite
  - MotionDetection, [58](#)
- color
  - ColorDetection.h, [86](#)
  - MultiColorLight, [74](#)
- ColorDetection, [12](#)

- beginAutoMode, 13
- configure, 13
- getAmbientLight, 14
- getColorValue, 14
- rgbwSensor, 15
- colorDetection
  - Dezibot, 18
- ColorDetection.h
  - AUTO, 87
  - color, 86
  - duration, 87
  - MANUAL, 87
  - MS1280, 87
  - MS160, 87
  - MS320, 87
  - MS40, 87
  - MS640, 87
  - MS80, 87
  - VEML\_BLUE, 86
  - VEML\_GREEN, 86
  - VEML\_RED, 86
  - VEML\_WHITE, 86
  - vemlMode, 87
- colorInverted
  - Display, 28
- colRange
  - DisplayCMDs.h, 98
- Communication, 15
  - begin, 15
  - onReceive, 16
  - sendMessage, 16
  - setGroupNumber, 16
- communication
  - Dezibot, 18
- Communication.cpp
  - changedConnectionCallback, 88
  - mesh, 90
  - newConnectionCallback, 88
  - nodeTimeAdjustedCallback, 89
  - userScheduler, 90
  - vTaskUpdate, 89
- Communication.h
  - MESH\_PASSWORD, 91
  - MESH\_PORT, 91
  - MESH\_PREFIX, 91
- completeOn
  - DisplayCMDs.h, 98
- configure
  - ColorDetection, 13
- correctionThreshold
  - Motion, 52
- currLine
  - Display, 28
- data\_byte
  - DisplayCMDs.h, 98
- DAYLIGHT
  - LightDetection.h, 107
- DEFAULT\_BASE\_VALUE
  - Motion.h, 110
- defaultShakeThreshold
  - MotionDetection, 67
- detection
  - Motion, 52
- Dezibot, 17
  - begin, 18
  - colorDetection, 18
  - communication, 18
  - Dezibot, 18
  - display, 19
  - infraredLight, 19
  - lightDetection, 19
  - motion, 19
  - multiColorLight, 19
- Dezibot.cpp
  - SCL\_PIN, 92
  - SDA\_PIN, 92
- Direction
  - MotionDetection.h, 121
- disableDisplay
  - DisplayCMDs.h, 99
- Display, 20
  - begin, 21
  - charsOnCurrLine, 28
  - clear, 21
  - colorInverted, 28
  - currLine, 28
  - flipOrientation, 22
  - invertColor, 22
  - orientationFlipped, 28
  - print, 23, 24
  - println, 25, 26
  - sendDisplayCMD, 26
  - stringToCharArray, 27
  - updateLine, 27
- display
  - Dezibot, 19
- DisplayCMDs.h
  - activateDisplay, 98
  - addressingMode, 98
  - cmd\_byte, 98
  - colRange, 98
  - completeOn, 98
  - data\_byte, 98
  - disableDisplay, 99
  - muxRatio, 99
  - pageRange, 99
  - setChargePump, 99
  - setComDirectionFlipped, 99
  - setComDirectionNormal, 99
  - setComHardwareConfig, 100
  - setContrast, 100
  - setInverseMode, 100
  - setNormalMode, 100
  - setOffset, 100
  - setOscFreq, 100
  - setSegmentMap, 101

- setSegmentReMap, 101
  - setStartLine, 101
  - stopCompleteOn, 101
- DL\_BOTTOM
  - LightDetection.h, 107
- DL\_FRONT
  - LightDetection.h, 107
- DL\_PT\_BOTTOM\_ADC
  - LightDetection, 44
- DL\_PT\_ENABLE
  - LightDetection, 44
- DL\_PT\_FRONT\_ADC
  - LightDetection, 44
- done
  - averageMeasurement, 11
- doxymain.md, 84
- duration
  - ColorDetection.h, 87
- duty
  - Motor, 71
- DUTY\_RES
  - Motion.h, 110
- enabled
  - VEML\_CONFIG, 81
- end
  - MotionDetection, 58
- Error
  - MotionDetection.h, 121
- example/advanced/Ampel1/Ampel1.ino, 84
- example/advanced/Ampel2/Ampel2.ino, 84
- example/advanced/Ampel3/Ampel3.ino, 84
- example/advanced/FindAFriend/FindAFriend.ino, 84
- example/advanced/FrequencyFindAFriend/FrequencyFindAFriend.ino, 84
- example/advanced/phototaxis/phototaxis.ino, 84
- example/advanced/simpleMorse/simpleMorse.ino, 84
- example/advanced/wuerfeln/wuerfeln.ino, 84
- example/advanced/zaehlen/zaehlen.ino, 84
- example/color\_detection/color\_detection.ino, 84
- example/display/basic/basic/basic.ino, 84
- example/Fernbedienung/empfaenger/empfaenger.ino, 84
- example/Fernbedienung/sender/sender.ino, 84
- example/IMU/Back\_to\_Origin/Back\_to\_Origin.ino, 84
- example/IMU/Detection\_Print/Detection\_Print.ino, 84
- example/IMU/Motion\_Correction/motion\_correction/motion\_correction.ino, 84
- example/IMU/Shake\_Detection/shake\_detection/shake\_detection.ino, 84
- example/IMU/Tilt\_Detection/tilt\_detection/tilt\_detection.ino, 84
- example/IMU/Upside\_Downside\_Detection/Upside\_Downside\_Detection.ino, 84
- example/IRDirection/IRDirection/IRDirection.ino, 84
- example/Led/ColorCycle/ColorCycle/ColorCycle.ino, 84
- example/lightdetection\_serial/lightdetection\_serial.ino, 84
- example/motion\_indicating/motion\_indicating.ino, 84
- example/motion\_minimum/motion\_minimum.ino, 84
- example/start/start.ino, 84
- exposureTime
  - VEML\_CONFIG, 81
- FIFO\_CONFIG1
  - IMU\_COMMANDS.h, 115
- FIFO\_CONFIG2
  - IMU\_COMMANDS.h, 115
- FIFO\_CONFIG5
  - IMU\_COMMANDS.h, 115
- FIFO\_COUNTH
  - IMU\_COMMANDS.h, 115
- FIFO\_COUNTL
  - IMU\_COMMANDS.h, 115
- FIFO\_DATA
  - IMU\_COMMANDS.h, 115
- FIFO\_Package, 29
  - accel, 29
  - gyro, 30
  - header, 30
  - temperature, 30
  - timestamp, 30
- flipOrientation
  - Display, 22
- Flipped
  - MotionDetection.h, 121
- font8x8\_colwise
  - CharTable.h, 94
- FREQUENCY
  - Motion.h, 110
- frequency
  - MotionDetection, 68
  - MotionDetection.h, 121
- front
  - InfraredLight, 37
- getAcceleration
  - MotionDetection, 59
- getAmbientLight
  - ColorDetection, 14
- getAverageValue
  - LightDetection, 40
- getBrightest
  - LightDetection, 42
- getColorValue
  - ColorDetection, 14
- getDataFromFIFO
  - MotionDetection, 59
- getRotation
  - MotionDetection, 60
- getSpeed
  - Motor, 69
- getTemperature
  - MotionDetection, 61
- getTilt
  - MotionDetection, 61
- getTiltDirection

- MotionDetection, [62](#)
- getValue
  - LightDetection, [42](#)
- getWhoAml
  - MotionDetection, [63](#)
- gForceCalib
  - MotionDetection, [68](#)
- gyro
  - FIFO\_Package, [30](#)
- GYRO\_DATA\_X\_HIGH
  - IMU\_COMMANDS.h, [116](#)
- GYRO\_DATA\_X\_LOW
  - IMU\_COMMANDS.h, [116](#)
- GYRO\_DATA\_Y\_HIGH
  - IMU\_COMMANDS.h, [116](#)
- GYRO\_DATA\_Y\_LOW
  - IMU\_COMMANDS.h, [116](#)
- GYRO\_DATA\_Z\_HIGH
  - IMU\_COMMANDS.h, [116](#)
- GYRO\_DATA\_Z\_LOW
  - IMU\_COMMANDS.h, [116](#)
- handler
  - MotionDetection, [68](#)
- header
  - FIFO\_Package, [30](#)
- IMU\_COMMANDS.h
  - ACCEL\_DATA\_X\_HIGH, [113](#)
  - ACCEL\_DATA\_X\_LOW, [113](#)
  - ACCEL\_DATA\_Y\_HIGH, [113](#)
  - ACCEL\_DATA\_Y\_LOW, [113](#)
  - ACCEL\_DATA\_Z\_HIGH, [113](#)
  - ACCEL\_DATA\_Z\_LOW, [114](#)
  - ADDR\_MASK, [114](#)
  - BLK\_SEL\_R, [114](#)
  - BLK\_SEL\_W, [114](#)
  - CMD\_READ, [114](#)
  - CMD\_WRITE, [114](#)
  - FIFO\_CONFIG1, [115](#)
  - FIFO\_CONFIG2, [115](#)
  - FIFO\_CONFIG5, [115](#)
  - FIFO\_COUNTH, [115](#)
  - FIFO\_COUNTL, [115](#)
  - FIFO\_DATA, [115](#)
  - GYRO\_DATA\_X\_HIGH, [116](#)
  - GYRO\_DATA\_X\_LOW, [116](#)
  - GYRO\_DATA\_Y\_HIGH, [116](#)
  - GYRO\_DATA\_Y\_LOW, [116](#)
  - GYRO\_DATA\_Z\_HIGH, [116](#)
  - GYRO\_DATA\_Z\_LOW, [116](#)
  - INTF\_CONFIG0, [117](#)
  - M\_R, [117](#)
  - M\_W, [117](#)
  - MADDR\_R, [117](#)
  - MADDR\_W, [117](#)
  - MCLK\_RDY, [117](#)
  - PWR\_MGMT0, [118](#)
  - REG\_TEMP\_HIGH, [118](#)
  - REG\_TEMP\_LOW, [118](#)
  - TMST\_CONFIG1, [118](#)
  - WHO\_AM\_I, [118](#)
- IMUResult, [30](#)
  - x, [31](#)
  - y, [31](#)
  - z, [31](#)
- InfraredLED, [31](#)
  - begin, [32](#)
  - channel, [35](#)
  - InfraredLED, [32](#)
  - ledPin, [35](#)
  - pwmChannel, [35](#)
  - pwmTimer, [35](#)
  - sendFrequency, [33](#)
  - setState, [33](#)
  - timer, [35](#)
  - turnOff, [34](#)
  - turnOn, [34](#)
- InfraredLED.cpp
  - pwmSpeedMode, [102](#)
- InfraredLight, [36](#)
  - begin, [37](#)
  - bottom, [37](#)
  - front, [37](#)
  - IRBottomPin, [38](#)
  - IRFrontPin, [38](#)
- infraredLight
  - Dezibot, [19](#)
- initFIFO
  - MotionDetection, [63](#)
- INTF\_CONFIG0
  - IMU\_COMMANDS.h, [117](#)
- invertColor
  - Display, [22](#)
- IR
  - LightDetection.h, [107](#)
- IR\_BACK
  - LightDetection.h, [107](#)
- IR\_FRONT
  - LightDetection.h, [107](#)
- IR\_LEFT
  - LightDetection.h, [107](#)
- IR\_PT\_BACK\_ADC
  - LightDetection, [44](#)
- IR\_PT\_ENABLE
  - LightDetection, [44](#)
- IR\_PT\_FRONT\_ADC
  - LightDetection, [45](#)
- IR\_PT\_LEFT\_ADC
  - LightDetection, [45](#)
- IR\_PT\_RIGHT\_ADC
  - LightDetection, [45](#)
- IR\_RIGHT
  - LightDetection.h, [107](#)
- IRBottomPin
  - InfraredLight, [38](#)
- IRFrontPin

- InfraredLight, [38](#)
- isShaken
  - MotionDetection, [64](#)
- ledAmount
  - MultiColorLight, [79](#)
- LEDC\_MODE
  - Motion.h, [111](#)
- ledPin
  - InfraredLED, [35](#)
  - MultiColorLight, [79](#)
- leds
  - MultiColorLight.h, [124](#)
- Left
  - MotionDetection.h, [121](#)
- left
  - Motion, [52](#)
- LEFT\_MOTOR\_DUTY
  - Motion, [53](#)
- leftMotorTask
  - Motion, [48](#)
- LightDetection, [38](#)
  - begin, [39](#)
  - beginDaylight, [40](#)
  - beginInfrared, [40](#)
  - DL\_PT\_BOTTOM\_ADC, [44](#)
  - DL\_PT\_ENABLE, [44](#)
  - DL\_PT\_FRONT\_ADC, [44](#)
  - getAverageValue, [40](#)
  - getBrightest, [42](#)
  - getValue, [42](#)
  - IR\_PT\_BACK\_ADC, [44](#)
  - IR\_PT\_ENABLE, [44](#)
  - IR\_PT\_FRONT\_ADC, [45](#)
  - IR\_PT\_LEFT\_ADC, [45](#)
  - IR\_PT\_RIGHT\_ADC, [45](#)
  - readDLPT, [43](#)
  - readIRPT, [43](#)
- lightDetection
  - Dezibot, [19](#)
- LightDetection.h
  - DAYLIGHT, [107](#)
  - DL\_BOTTOM, [107](#)
  - DL\_FRONT, [107](#)
  - IR, [107](#)
  - IR\_BACK, [107](#)
  - IR\_FRONT, [107](#)
  - IR\_LEFT, [107](#)
  - IR\_RIGHT, [107](#)
  - photoTransistors, [107](#)
  - ptType, [107](#)
- M\_R
  - IMU\_CMDs.h, [117](#)
- M\_W
  - IMU\_CMDs.h, [117](#)
- MADDR\_R
  - IMU\_CMDs.h, [117](#)
- MADDR\_W
  - IMU\_CMDs.h, [117](#)
- IMU\_CMDs.h, [117](#)
- MANUAL
  - ColorDetection.h, [87](#)
- maxBrightness
  - MultiColorLight, [79](#)
- MCLK\_RDY
  - IMU\_CMDs.h, [117](#)
- measurementAmount
  - averageMeasurement, [11](#)
- mesh
  - Communication.cpp, [90](#)
- MESH\_PASSWORD
  - Communication.h, [91](#)
- MESH\_PORT
  - Communication.h, [91](#)
- MESH\_PREFIX
  - Communication.h, [91](#)
- mode
  - VEML\_CONFIG, [81](#)
- Motion, [46](#)
  - begin, [47](#)
  - buffer, [52](#)
  - correctionThreshold, [52](#)
  - detection, [52](#)
  - left, [52](#)
  - LEFT\_MOTOR\_DUTY, [53](#)
  - leftMotorTask, [48](#)
  - MOTOR\_LEFT\_PIN, [53](#)
  - MOTOR\_RIGHT\_PIN, [53](#)
  - move, [48](#)
  - moveTask, [49](#)
  - moveWithoutCorrection, [49](#)
  - right, [53](#)
  - RIGHT\_MOTOR\_DUTY, [53](#)
  - rightMotorTask, [50](#)
  - rotateAntiClockwise, [50](#)
  - rotateClockwise, [51](#)
  - stop, [51](#)
  - xAntiClockwiseTaskHandle, [53](#)
  - xClockwiseTaskHandle, [54](#)
  - xLastWakeTime, [54](#)
  - xMoveTaskHandle, [54](#)
- motion
  - Dezibot, [19](#)
- Motion.h
  - CHANNEL\_LEFT, [110](#)
  - CHANNEL\_RIGHT, [110](#)
  - DEFAULT\_BASE\_VALUE, [110](#)
  - DUTY\_RES, [110](#)
  - FREQUENCY, [110](#)
  - LEDC\_MODE, [111](#)
  - TIMER, [111](#)
- MotionDetection, [54](#)
  - begin, [56](#)
  - buf, [67](#)
  - bufferLength, [67](#)
  - calibrateZAxis, [57](#)
  - cmdRead, [58](#)

- cmdWrite, [58](#)
- defaultShakeThreshold, [67](#)
- end, [58](#)
- frequency, [68](#)
- getAcceleration, [59](#)
- getDataFromFIFO, [59](#)
- getRotation, [60](#)
- getTemperature, [61](#)
- getTilt, [61](#)
- getTiltDirection, [62](#)
- getWhoAmI, [63](#)
- gForceCalib, [68](#)
- handler, [68](#)
- initFIFO, [63](#)
- isShaken, [64](#)
- MotionDetection, [56](#)
- MREG1, [56](#)
- MREG2, [56](#)
- MREG3, [56](#)
- readDoubleRegister, [64](#)
- readFromRegisterBank, [64](#)
- readRegister, [65](#)
- registerBank, [56](#)
- resetRegisterBankAccess, [65](#)
- writeRegister, [66](#)
- writeToRegisterBank, [66](#)
- MotionDetection.h
  - Axis, [121](#)
  - Back, [121](#)
  - Direction, [121](#)
  - Error, [121](#)
  - Flipped, [121](#)
  - Front, [121](#)
  - Left, [121](#)
  - Neutral, [121](#)
  - Right, [121](#)
  - xAxis, [121](#)
  - yAxis, [121](#)
  - zAxis, [121](#)
- Motor, [68](#)
  - begin, [69](#)
  - channel, [71](#)
  - duty, [71](#)
  - getSpeed, [69](#)
  - Motor, [69](#)
  - pin, [71](#)
  - setSpeed, [70](#)
  - timer, [72](#)
- MOTOR\_LEFT\_PIN
  - Motion, [53](#)
- MOTOR\_RIGHT\_PIN
  - Motion, [53](#)
- move
  - Motion, [48](#)
- moveTask
  - Motion, [49](#)
- moveWithoutCorrection
  - Motion, [49](#)
- MREG1
  - MotionDetection, [56](#)
- MREG2
  - MotionDetection, [56](#)
- MREG3
  - MotionDetection, [56](#)
- MS1280
  - ColorDetection.h, [87](#)
- MS160
  - ColorDetection.h, [87](#)
- MS320
  - ColorDetection.h, [87](#)
- MS40
  - ColorDetection.h, [87](#)
- MS640
  - ColorDetection.h, [87](#)
- MS80
  - ColorDetection.h, [87](#)
- MultiColorLight, [72](#)
  - begin, [73](#)
  - blink, [73](#)
  - color, [74](#)
  - ledAmount, [79](#)
  - ledPin, [79](#)
  - maxBrightness, [79](#)
  - MultiColorLight, [73](#)
  - rgbLeds, [79](#)
  - setLed, [75](#), [76](#)
  - setTopLeds, [77](#)
  - turnOffLed, [78](#)
- multiColorLight
  - Dezibot, [19](#)
- MultiColorLight.h
  - ALL, [124](#)
  - BOTTOM, [124](#)
  - leds, [124](#)
  - TOP, [124](#)
  - TOP\_LEFT, [124](#)
  - TOP\_RIGHT, [124](#)
- muxRatio
  - DisplayCMDs.h, [99](#)
- Neutral
  - MotionDetection.h, [121](#)
- newConnectionCallback
  - Communication.cpp, [88](#)
- nodeTimeAdjustedCallback
  - Communication.cpp, [89](#)
- onReceive
  - Communication, [16](#)
- Orientation, [80](#)
  - xRotation, [80](#)
  - yRotation, [80](#)
- orientationFlipped
  - Display, [28](#)
- pageRange
  - DisplayCMDs.h, [99](#)

- photoTransistors
  - LightDetection.h, [107](#)
- pin
  - Motor, [71](#)
- print
  - Display, [23](#), [24](#)
- println
  - Display, [25](#), [26](#)
- ptType
  - LightDetection.h, [107](#)
- pwmChannel
  - InfraredLED, [35](#)
- pwmSpeedMode
  - InfraredLED.cpp, [102](#)
- pwmTimer
  - InfraredLED, [35](#)
- PWR\_MGMT0
  - IMU\_COMMANDS.h, [118](#)
- readDLPT
  - LightDetection, [43](#)
- readDoubleRegister
  - MotionDetection, [64](#)
- readFromRegisterBank
  - MotionDetection, [64](#)
- readIRPT
  - LightDetection, [43](#)
- README.md, [84](#)
- readRegister
  - MotionDetection, [65](#)
- REG\_TEMP\_HIGH
  - IMU\_COMMANDS.h, [118](#)
- REG\_TEMP\_LOW
  - IMU\_COMMANDS.h, [118](#)
- registerBank
  - MotionDetection, [56](#)
- resetRegisterBankAccess
  - MotionDetection, [65](#)
- result
  - averageMeasurement, [12](#)
- rgbLeds
  - MultiColorLight, [79](#)
- rgbwSensor
  - ColorDetection, [15](#)
- Right
  - MotionDetection.h, [121](#)
- right
  - Motion, [53](#)
- RIGHT\_MOTOR\_DUTY
  - Motion, [53](#)
- rightMotorTask
  - Motion, [50](#)
- rotateAntiClockwise
  - Motion, [50](#)
- rotateClockwise
  - Motion, [51](#)
- SCL\_PIN
  - Dezibot.cpp, [92](#)
- SDA\_PIN
  - Dezibot.cpp, [92](#)
- sendDisplayCMD
  - Display, [26](#)
- sendFrequency
  - InfraredLED, [33](#)
- sendMessage
  - Communication, [16](#)
- sensor
  - averageMeasurement, [12](#)
- setChargePump
  - DisplayCOMMANDS.h, [99](#)
- setComDirectionFlipped
  - DisplayCOMMANDS.h, [99](#)
- setComDirectionNormal
  - DisplayCOMMANDS.h, [99](#)
- setComHardwareConfig
  - DisplayCOMMANDS.h, [100](#)
- setContrast
  - DisplayCOMMANDS.h, [100](#)
- setGroupNumber
  - Communication, [16](#)
- setInverseMode
  - DisplayCOMMANDS.h, [100](#)
- setLed
  - MultiColorLight, [75](#), [76](#)
- setNormalMode
  - DisplayCOMMANDS.h, [100](#)
- setOffset
  - DisplayCOMMANDS.h, [100](#)
- setOscFreq
  - DisplayCOMMANDS.h, [100](#)
- setSegmentMap
  - DisplayCOMMANDS.h, [101](#)
- setSegmentReMap
  - DisplayCOMMANDS.h, [101](#)
- setSpeed
  - Motor, [70](#)
- setStartLine
  - DisplayCOMMANDS.h, [101](#)
- setState
  - InfraredLED, [33](#)
- setTopLeds
  - MultiColorLight, [77](#)
- src/colorDetection/ColorDetection.cpp, [84](#)
- src/colorDetection/ColorDetection.h, [85](#)
- src/communication/Communication.cpp, [87](#)
- src/communication/Communication.h, [90](#)
- src/Dezibot.cpp, [92](#)
- src/Dezibot.h, [93](#)
- src/display/CharTable.h, [94](#)
- src/display/Display.cpp, [95](#)
- src/display/Display.h, [96](#)
- src/display/DisplayCOMMANDS.h, [97](#)
- src/infraredLight/InfraredLED.cpp, [102](#)
- src/infraredLight/InfraredLight.cpp, [103](#)
- src/infraredLight/InfraredLight.h, [103](#)
- src/lightDetection/LightDetection.cpp, [105](#)

src/lightDetection/LightDetection.h, 105  
 src/motion/Motion.cpp, 107  
 src/motion/Motion.h, 108  
 src/motion/Motor.cpp, 111  
 src/motionDetection/IMU\_CMDs.h, 112  
 src/motionDetection/MotionDetection.cpp, 119  
 src/motionDetection/MotionDetection.h, 119  
 src/multiColorLight/ColorConstants.h, 122  
 src/multiColorLight/MultiColorLight.cpp, 122  
 src/multiColorLight/MultiColorLight.h, 123  
 stop  
     Motion, 51  
 stopCompleteOn  
     DisplayCMDs.h, 101  
 stringToCharArray  
     Display, 27  
  
 temperature  
     FIFO\_Package, 30  
 timeBetween  
     averageMeasurement, 12  
 TIMER  
     Motion.h, 111  
 timer  
     InfraredLED, 35  
     Motor, 72  
 timestamp  
     FIFO\_Package, 30  
 TMST\_CONFIG1  
     IMU\_CMDs.h, 118  
 TOP  
     MultiColorLight.h, 124  
 TOP\_LEFT  
     MultiColorLight.h, 124  
 TOP\_RIGHT  
     MultiColorLight.h, 124  
 turnOff  
     InfraredLED, 34  
 turnOffLed  
     MultiColorLight, 78  
 turnOn  
     InfraredLED, 34  
  
 updateLine  
     Display, 27  
 userScheduler  
     Communication.cpp, 90  
  
 VEML\_BLUE  
     ColorDetection.h, 86  
 VEML\_CONFIG, 81  
     enabled, 81  
     exposureTime, 81  
     mode, 81  
 VEML\_GREEN  
     ColorDetection.h, 86  
 VEML\_RED  
     ColorDetection.h, 86  
 VEML\_WHITE  
     ColorDetection.h, 86  
  
 vemlMode  
     ColorDetection.h, 87  
 vTaskUpdate  
     Communication.cpp, 89  
  
 WHO\_AM\_I  
     IMU\_CMDs.h, 118  
 writeRegister  
     MotionDetection, 66  
 writeToRegisterBank  
     MotionDetection, 66  
  
 x  
     IMUResult, 31  
 xAntiClockwiseTaskHandle  
     Motion, 53  
 xAxis  
     MotionDetection.h, 121  
 xClockwiseTaskHandle  
     Motion, 54  
 xLastWakeTime  
     Motion, 54  
 xMoveTaskHandle  
     Motion, 54  
 xRotation  
     Orientation, 80  
  
 y  
     IMUResult, 31  
 yAxis  
     MotionDetection.h, 121  
 yRotation  
     Orientation, 80  
  
 z  
     IMUResult, 31  
 zAxis  
     MotionDetection.h, 121