

## Prova Prática - Laboratório de Processadores

Generated by Doxygen 1.9.1



<b>1 Prova Prática de Laboratório de Processadores</b>	<b>1</b>
1.1 Como Executar	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 AdcConfig Struct Reference	7
4.1.1 Member Data Documentation	7
4.1.1.1 adc_number	8
4.1.1.2 channels	8
4.1.1.3 gpio	8
4.1.1.4 mode	8
4.1.1.5 prescaler	8
4.1.1.6 rcc_clock	8
4.1.1.7 rcc_reset	8
4.1.1.8 resolution	8
4.1.1.9 sample_time	9
4.2 ButterworthFilter Class Reference	9
4.2.1 Detailed Description	9
4.2.2 Constructor & Destructor Documentation	9
4.2.2.1 ButterworthFilter()	9
4.2.3 Member Function Documentation	10
4.2.3.1 update()	10
4.3 Button Class Reference	10
4.3.1 Constructor & Destructor Documentation	10
4.3.1.1 Button()	11
4.3.2 Member Function Documentation	12
4.3.2.1 get_status()	12
4.4 ClockConfig Struct Reference	12
4.4.1 Member Data Documentation	12
4.4.1.1 clock_scale	12
4.4.1.2 clocksource	13
4.4.1.3 reload	13
4.5 GpioConfig Struct Reference	13
4.5.1 Member Data Documentation	13
4.5.1.1 alt_func_num	13
4.5.1.2 mode	13
4.5.1.3 otype	14
4.5.1.4 pin	14

4.5.1.5 port	14
4.5.1.6 pull_resistor	14
4.5.1.7 rcc_clock	14
4.5.1.8 speed	14
4.6 HalAdc< number_of_channels > Class Template Reference	14
4.6.1 Constructor & Destructor Documentation	15
4.6.1.1 HalAdc()	15
4.6.2 Member Function Documentation	15
4.6.2.1 get_adc_reading()	15
4.6.2.2 update_reading()	16
4.7 HalGpio Class Reference	16
4.7.1 Constructor & Destructor Documentation	16
4.7.1.1 HalGpio()	16
4.7.2 Member Function Documentation	17
4.7.2.1 read()	17
4.7.2.2 toggle()	17
4.7.2.3 write()	17
4.8 HalPwm Class Reference	17
4.8.1 Constructor & Destructor Documentation	18
4.8.1.1 HalPwm()	18
4.8.2 Member Function Documentation	18
4.8.2.1 set_compare()	18
4.9 HalTimer Class Reference	18
4.9.1 Constructor & Destructor Documentation	19
4.9.1.1 HalTimer()	19
4.9.2 Member Function Documentation	19
4.9.2.1 get_time()	19
4.9.2.2 increment_system_ticks()	20
4.9.2.3 reset()	20
4.9.2.4 sleep()	20
4.10 LineSensors< number_of_sensors > Class Template Reference	20
4.10.1 Constructor & Destructor Documentation	21
4.10.1.1 LineSensors()	21
4.10.2 Member Function Documentation	21
4.10.2.1 calibrate_black()	21
4.10.2.2 calibrate_white()	21
4.10.2.3 get_position()	21
4.11 Locomotion Class Reference	22
4.11.1 Constructor & Destructor Documentation	22
4.11.1.1 Locomotion()	22
4.11.2 Member Function Documentation	22
4.11.2.1 linear_decay()	22

4.11.2.2 set_speeds()	23
4.12 Motor Class Reference	23
4.12.1 Constructor & Destructor Documentation	23
4.12.1.1 Motor()	24
4.12.2 Member Function Documentation	24
4.12.2.1 set_speed()	24
4.13 MotorConfig Struct Reference	24
4.13.1 Member Data Documentation	25
4.13.1.1 backward_pwm	25
4.13.1.2 forward_pwm	25
4.14 PidController Class Reference	26
4.14.1 Detailed Description	26
4.14.2 Constructor & Destructor Documentation	26
4.14.2.1 PidController()	26
4.14.3 Member Function Documentation	27
4.14.3.1 reset()	27
4.14.3.2 set_parameters()	27
4.14.3.3 set_setpoint()	27
4.14.3.4 update() [1/2]	28
4.14.3.5 update() [2/2]	28
4.15 PwmConfig Struct Reference	28
4.15.1 Member Data Documentation	29
4.15.1.1 clock_div	29
4.15.1.2 gpio	29
4.15.1.3 oc_id	29
4.15.1.4 oc_mode	30
4.15.1.5 period	30
4.15.1.6 prescaler	30
4.15.1.7 rcc_clock	30
4.15.1.8 timer	30
<b>5 File Documentation</b>	<b>31</b>
5.1 cfg/constants.hpp File Reference	31
5.1.1 Variable Documentation	31
5.1.1.1 filter_frequency	32
5.1.1.2 kd	32
5.1.1.3 ki	32
5.1.1.4 kp	32
5.1.1.5 left_deadzone	32
5.1.1.6 linear_base_speed	32
5.1.1.7 linear_decay	32
5.1.1.8 max_integral	32

5.1.1.9 right_deadzone . . . . .	33
5.1.1.10 saturation . . . . .	33
5.2 cfg/target.hpp File Reference . . . . .	33
5.2.1 Variable Documentation . . . . .	34
5.2.1.1 adc_channels . . . . .	34
5.2.1.2 adc_num_channels . . . . .	34
5.2.1.3 adc_readings_per_channel . . . . .	34
5.2.1.4 button_config . . . . .	35
5.2.1.5 button_pull_resistor . . . . .	35
5.2.1.6 clock_config . . . . .	35
5.2.1.7 led_config . . . . .	35
5.2.1.8 left_motor_config . . . . .	35
5.2.1.9 line_sensors_config . . . . .	36
5.2.1.10 right_motor_config . . . . .	36
5.3 inc/butterworth_filter.hpp File Reference . . . . .	36
5.4 inc/hal/hal_adc.hpp File Reference . . . . .	37
5.5 inc/hal/hal_gpio.hpp File Reference . . . . .	38
5.6 inc/hal/hal_pwm.hpp File Reference . . . . .	39
5.7 inc/hal/hal_timer.hpp File Reference . . . . .	40
5.8 inc/mcu.hpp File Reference . . . . .	40
5.8.1 Function Documentation . . . . .	41
5.8.1.1 mcu_init() . . . . .	41
5.9 inc/pid_controller.hpp File Reference . . . . .	42
5.10 inc/proxy/button.hpp File Reference . . . . .	43
5.10.1 Enumeration Type Documentation . . . . .	44
5.10.1.1 button_pull_resistor_t . . . . .	44
5.10.1.2 button_status_t . . . . .	44
5.11 inc/proxy/line_sensors.hpp File Reference . . . . .	44
5.12 inc/proxy/locomotion.hpp File Reference . . . . .	45
5.13 inc/proxy/motor.hpp File Reference . . . . .	46
5.13.1 Variable Documentation . . . . .	47
5.13.1.1 max_motors_speed . . . . .	47
5.13.1.2 min_motors_speed . . . . .	47
5.14 inc/utls.hpp File Reference . . . . .	48
5.14.1 Macro Definition Documentation . . . . .	48
5.14.1.1 constrain . . . . .	48
5.14.1.2 map . . . . .	48
5.15 README.md File Reference . . . . .	49
5.16 src/butterworth_filter.cpp File Reference . . . . .	49
5.16.1 Variable Documentation . . . . .	49
5.16.1.1 sqrt2 . . . . .	49
5.17 src/hal/hal_adc.cpp File Reference . . . . .	50

5.17.1 Macro Definition Documentation . . . . .	50
5.17.1.1 __HAL_ADC_CPP__ . . . . .	50
5.18 src/hal/hal_gpio.cpp File Reference . . . . .	50
5.19 src/hal/hal_pwm.cpp File Reference . . . . .	51
5.20 src/hal/hal_timer.cpp File Reference . . . . .	51
5.20.1 Function Documentation . . . . .	52
5.20.1.1 sys_tick_handler() . . . . .	52
5.21 src/main.cpp File Reference . . . . .	52
5.21.1 Function Documentation . . . . .	52
5.21.1.1 main() . . . . .	53
5.21.2 Variable Documentation . . . . .	53
5.21.2.1 angular_command . . . . .	53
5.21.2.2 angular_position . . . . .	53
5.21.2.3 line_measure . . . . .	53
5.21.2.4 linear_command . . . . .	53
5.21.2.5 stopped . . . . .	53
5.22 src/mcu.cpp File Reference . . . . .	54
5.22.1 Function Documentation . . . . .	54
5.22.1.1 mcu_init() . . . . .	54
5.23 src/pid_controller.cpp File Reference . . . . .	54
5.24 src/proxy/button.cpp File Reference . . . . .	55
5.25 src/proxy/line_sensors.cpp File Reference . . . . .	55
5.25.1 Macro Definition Documentation . . . . .	56
5.25.1.1 __LINE_SENSORS_CPP__ . . . . .	56
5.25.2 Variable Documentation . . . . .	56
5.25.2.1 default_black_value . . . . .	56
5.25.2.2 default_white_value . . . . .	56
5.26 src/proxy/locomotion.cpp File Reference . . . . .	57
5.27 src/proxy/motor.cpp File Reference . . . . .	57

<b>Index</b>	<b>59</b>
--------------	-----------





## Chapter 1

# Prova Prática de Laboratório de Processadores

Este projeto consiste em uma biblioteca HAL em C++ para embarcados e sua aplicação em um robô seguidor de linha, visando facilitar futuras implementações que utilizam as classes definidas, que incluem funcionalidades do microcontrolador, sensores e atuadores.

### 1.1 Como Executar

1. Inicialmente, o repositório deve ser clonado localmente, em seguida, devem ser executados os seguintes comandos:
2. `git submodule update --init`
3. `make -C lib/libopencm3`
4. `make`



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">AdcConfig</a> . . . . .	7
<a href="#">ButterworthFilter</a>	
Implementation of Butterworth second order low-pass filter A generic digital filter follows the relation $a_0 * y[k] = \sum(b_i * x[k - i]) - \sum(a_j * y[k - j])$ Where $x[k]$ - measurement at instant $k$ $y[k]$ - filtered signal at instant $k$ The Butterworth filter have the special property of being a maximally flat magnitude filter, in other words, is the best filter that doesn't present distortions around the cutoff frequency The formula for the continuous coefficients of the Butterworth filter is available here: <a href="https://en.wikipedia.org/wiki/Butterworth_filter">https://en.wikipedia.org/wiki/Butterworth_filter</a> The discrete version were computed with the Tustin method: <a href="https://en.wikipedia.org/wiki/Bilinear_transform">https://en.wikipedia.org/wiki/Bilinear_transform</a> . . . . .	9
<a href="#">Button</a> . . . . .	10
<a href="#">ClockConfig</a> . . . . .	12
<a href="#">GpioConfig</a> . . . . .	13
<a href="#">HalAdc&lt; number_of_channels &gt;</a> . . . . .	14
<a href="#">HalGpio</a> . . . . .	16
<a href="#">HalPwm</a> . . . . .	17
<a href="#">HalTimer</a> . . . . .	18
<a href="#">LineSensors&lt; number_of_sensors &gt;</a> . . . . .	20
<a href="#">Locomotion</a> . . . . .	22
<a href="#">Motor</a> . . . . .	23
<a href="#">MotorConfig</a> . . . . .	24
<a href="#">PidController</a>	
Implementation of simple PID controller Response = $K_p(\text{error} + K_i * \text{integral}(\text{error}) K_d * \frac{d}{dt}(\text{error}))$ . . . . .	26
<a href="#">PwmConfig</a> . . . . .	28



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

cfg/	constants.hpp	31
cfg/	target.hpp	33
inc/	butterworth_filter.hpp	36
inc/	mcu.hpp	40
inc/	pid_controller.hpp	42
inc/	utils.hpp	48
inc/hal/	hal_adc.hpp	37
inc/hal/	hal_gpio.hpp	38
inc/hal/	hal_pwm.hpp	39
inc/hal/	hal_timer.hpp	40
inc/proxy/	button.hpp	43
inc/proxy/	line_sensors.hpp	44
inc/proxy/	locomotion.hpp	45
inc/proxy/	motor.hpp	46
src/	butterworth_filter.cpp	49
src/	main.cpp	52
src/	mcu.cpp	54
src/	pid_controller.cpp	54
src/hal/	hal_adc.cpp	50
src/hal/	hal_gpio.cpp	50
src/hal/	hal_pwm.cpp	51
src/hal/	hal_timer.cpp	51
src/proxy/	button.cpp	55
src/proxy/	line_sensors.cpp	55
src/proxy/	locomotion.cpp	57
src/proxy/	motor.cpp	57



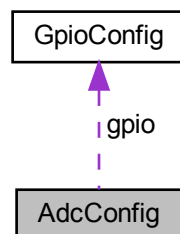
## Chapter 4

# Class Documentation

### 4.1 AdcConfig Struct Reference

```
#include <hal_adc.hpp>
```

Collaboration diagram for AdcConfig:



#### Public Attributes

- [GpioConfig gpio](#)
- [uint32\\_t adc\\_number](#)
- [uint32\\_t mode](#)
- [rcc\\_periph\\_clken rcc\\_clock](#)
- [rcc\\_periph\\_rst rcc\\_reset](#)
- [uint32\\_t prescaler](#)
- [uint32\\_t resolution](#)
- [uint8\\_t \\* channels](#)
- [uint8\\_t sample\\_time](#)

#### 4.1.1 Member Data Documentation

#### 4.1.1.1 adc\_number

```
uint32_t AdcConfig::adc_number
```

#### 4.1.1.2 channels

```
uint8_t* AdcConfig::channels
```

#### 4.1.1.3 gpio

```
GpioConfig AdcConfig::gpio
```

#### 4.1.1.4 mode

```
uint32_t AdcConfig::mode
```

#### 4.1.1.5 prescaler

```
uint32_t AdcConfig::prescaler
```

#### 4.1.1.6 rcc\_clock

```
rcc_periph_clken AdcConfig::rcc_clock
```

#### 4.1.1.7 rcc\_reset

```
rcc_periph_rst AdcConfig::rcc_reset
```

#### 4.1.1.8 resolution

```
uint32_t AdcConfig::resolution
```



#### 4.1.1.9 sample\_time

```
uint8_t AdcConfig::sample_time
```

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

- [inc/hal/hal\\_adc.hpp](#)

## 4.2 ButterworthFilter Class Reference

Implementation of Butterworth second order low-pass filter A generic digital filter follows the relation  $a_0 * y[k] = \sum(b_i * x[k - i]) - \sum(a_j * y[k - j])$  Where  $x[k]$  - measurement at instant  $k$   $y[k]$  - filtered signal at instant  $k$  The Butterworth filter have the special property of being a maximally flat magnitude filter, in other words, is the best filter that doesn't present distortions around the cutoff frequency The formula for the continuous coefficients of the Butterworth filter is available here: [https://en.wikipedia.org/wiki/Butterworth\\_filter](https://en.wikipedia.org/wiki/Butterworth_filter) The discrete version were computed with the Tustin method: [https://en.wikipedia.org/wiki/Bilinear\\_transform](https://en.wikipedia.org/wiki/Bilinear_transform).

```
#include <butterworth_filter.hpp>
```

### Public Member Functions

- [ButterworthFilter](#) (float cutoff\_frequency, float sampling\_frequency=1.0)  
*Construct a new Butterworth Second Order filter object.*
- float [update](#) (float x0)  
*Produces a new value from measured data.*

### 4.2.1 Detailed Description

Implementation of Butterworth second order low-pass filter A generic digital filter follows the relation  $a_0 * y[k] = \sum(b_i * x[k - i]) - \sum(a_j * y[k - j])$  Where  $x[k]$  - measurement at instant  $k$   $y[k]$  - filtered signal at instant  $k$  The Butterworth filter have the special property of being a maximally flat magnitude filter, in other words, is the best filter that doesn't present distortions around the cutoff frequency The formula for the continuous coefficients of the Butterworth filter is available here: [https://en.wikipedia.org/wiki/Butterworth\\_filter](https://en.wikipedia.org/wiki/Butterworth_filter) The discrete version were computed with the Tustin method: [https://en.wikipedia.org/wiki/Bilinear\\_transform](https://en.wikipedia.org/wiki/Bilinear_transform).

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 ButterworthFilter()

```
ButterworthFilter::ButterworthFilter (
    float cutoff_frequency,
    float sampling_frequency = 1.0 )
```

Construct a new Butterworth Second Order filter object.

**Parameters**

<i>cutoff_frequency</i>	Low-pass cutoff frequency in Hz
<i>sampling_frequency</i>	Sampling frequency in Hz.

## 4.2.3 Member Function Documentation

### 4.2.3.1 update()

```
float ButterworthFilter::update (
    float x0 )
```

Produces a new value from measured data.

**Parameters**

<i>x0</i>	Last measure
-----------	--------------

**Returns**

Filtered value

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

- [inc/butterworth\\_filter.hpp](#)
- [src/butterworth\\_filter.cpp](#)

## 4.3 Button Class Reference

```
#include <button.hpp>
```

**Public Member Functions**

- [Button](#) (const [GpioConfig](#) &gpio\_config, [button\\_pull\\_resistor\\_t](#) pull\_resistor)  
*Construct a new [Button](#) object.*
- [button\\_status\\_t](#) [get\\_status](#) ()  
*Provides the status of the chosen button.*

### 4.3.1 Constructor & Destructor Documentation

#### 4.3.1.1 Button()

```
Button::Button (
    const GpioConfig & gpio_config,
    button\_pull\_resistor\_t pull_resistor )
```

Construct a new [Button](#) object.

**Parameters**

<i>pio_config</i>	Configuration of the button GPIO port
<i>pull_resistor</i>	Type of pull resistor configuration

## 4.3.2 Member Function Documentation

### 4.3.2.1 get\_status()

```
button_status_t Button::get_status ( )
```

Provides the status of the chosen button.

**Returns**

Status of the button.

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

- inc/proxy/[button.hpp](#)
- src/proxy/[button.cpp](#)

## 4.4 ClockConfig Struct Reference

```
#include <mcu.hpp>
```

**Public Attributes**

- const struct rcc\_clock\_scale \* [clock\\_scale](#)
- uint32\_t [reload](#)
- uint8\_t [clocksource](#)

### 4.4.1 Member Data Documentation

#### 4.4.1.1 clock\_scale

```
const struct rcc_clock_scale* ClockConfig::clock_scale
```

#### 4.4.1.2 clocksource

```
uint8_t ClockConfig::clocksource
```

#### 4.4.1.3 reload

```
uint32_t ClockConfig::reload
```

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

- [inc/mcu.hpp](#)

## 4.5 GpioConfig Struct Reference

```
#include <hal_gpio.hpp>
```

### Public Attributes

- uint32\_t [port](#)
- uint16\_t [pin](#)
- uint8\_t [mode](#)
- uint8\_t [pull\\_resistor](#)
- rcc\_periph\_clken [rcc\\_clock](#)
- uint8\_t [otype](#)
- uint8\_t [speed](#)
- uint8\_t [alt\\_func\\_num](#)

### 4.5.1 Member Data Documentation

#### 4.5.1.1 alt\_func\_num

```
uint8_t GpioConfig::alt_func_num
```

#### 4.5.1.2 mode

```
uint8_t GpioConfig::mode
```

#### 4.5.1.3 otype

```
uint8_t GpioConfig::otype
```

#### 4.5.1.4 pin

```
uint16_t GpioConfig::pin
```

#### 4.5.1.5 port

```
uint32_t GpioConfig::port
```

#### 4.5.1.6 pull\_resistor

```
uint8_t GpioConfig::pull_resistor
```

#### 4.5.1.7 rcc\_clock

```
rcc_periph_clken GpioConfig::rcc_clock
```

#### 4.5.1.8 speed

```
uint8_t GpioConfig::speed
```

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

- [inc/hal/hal\\_gpio.hpp](#)

## 4.6 HalAdc< number\_of\_channels > Class Template Reference

```
#include <hal_adc.hpp>
```

## Public Member Functions

- [HalAdc](#) (const [AdcConfig](#) &adc\_config)  
*Construct a new Hal Adc object.*
- void [update\\_reading](#) ()  
*Update the ADC reading.*
- uint32\_t [get\\_adc\\_reading](#) (uint8\_t channel) const  
*Get the reading of the ADC.*

## 4.6.1 Constructor & Destructor Documentation

### 4.6.1.1 HalAdc()

```
template<uint8_t number_of_channels>
HalAdc< number_of_channels >::HalAdc (
    const AdcConfig & adc_config )
```

Construct a new Hal Adc object.

#### Parameters

<i>adc_config</i>	Configuration of the ADC
-------------------	--------------------------

## 4.6.2 Member Function Documentation

### 4.6.2.1 get\_adc\_reading()

```
template<uint8_t number_of_channels>
uint32_t HalAdc< number_of_channels >::get_adc_reading (
    uint8_t channel ) const
```

Get the reading of the ADC.

#### Parameters

<i>channel</i>	Channel of the ADC
----------------	--------------------

#### Returns

uint32\_t Reading of the ADC channel

#### 4.6.2.2 update\_reading()

```
template<uint8_t number_of_channels>
void HalAdc< number_of_channels >::update_reading (
    void )
```

Update the ADC reading.

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

- inc/hal/hal\_adc.hpp
- src/hal/hal\_adc.cpp

## 4.7 HalGpio Class Reference

```
#include <hal_gpio.hpp>
```

### Public Member Functions

- [HalGpio](#) (const [GpioConfig](#) &gpio\_config)  
*Construct a new Hal GPIO object.*
- bool [read](#) () const  
*Read the GPIO pin.*
- void [write](#) (bool pin\_state)  
*Write to the GPIO pin.*
- void [toggle](#) ()  
*Toggle the GPIO pin.*

### 4.7.1 Constructor & Destructor Documentation

#### 4.7.1.1 HalGpio()

```
HalGpio::HalGpio (
    const GpioConfig & gpio_config )
```

Construct a new Hal GPIO object.

#### Parameters

<i>gpio_config</i>	Configuration of the gpio instance
--------------------	------------------------------------



## 4.7.2 Member Function Documentation

### 4.7.2.1 read()

```
bool HalGpio::read ( ) const
```

Read the GPIO pin.

#### Returns

True if the pin is high, false otherwise

### 4.7.2.2 toggle()

```
void HalGpio::toggle ( )
```

Toggle the GPIO pin.

### 4.7.2.3 write()

```
void HalGpio::write (
    bool pin_state )
```

Write to the GPIO pin.

#### Parameters

<i>pin_state</i>	State of the GPIO pin
------------------	-----------------------

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

- [inc/hal/hal\\_gpio.hpp](#)
- [src/hal/hal\\_gpio.cpp](#)

## 4.8 HalPwm Class Reference

```
#include <hal_pwm.hpp>
```

## Public Member Functions

- [HalPwm](#) (const [PwmConfig](#) &pwm\_config)  
*Construct a new Hal Pwm object.*
- void [set\\_compare](#) (uint32\_t compare)  
*Set the PWM duty cycle.*

## 4.8.1 Constructor & Destructor Documentation

### 4.8.1.1 HalPwm()

```
HalPwm::HalPwm (
    const PwmConfig & pwm_config )
```

Construct a new Hal Pwm object.

#### Parameters

<a href="#">pwm_config</a>	Configuration for the pwm instance
----------------------------	------------------------------------

## 4.8.2 Member Function Documentation

### 4.8.2.1 set\_compare()

```
void HalPwm::set_compare (
    uint32_t compare )
```

Set the PWM duty cycle.

#### Parameters

<a href="#">compare</a>	Value to set the duty cycle
-------------------------	-----------------------------

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

- inc/hal/[hal\\_pwm.hpp](#)
- src/hal/[hal\\_pwm.cpp](#)

## 4.9 HalTimer Class Reference

```
#include <hal_timer.hpp>
```

## Public Member Functions

- [HalTimer](#) ()  
*Construct a new Hal Timer object.*
- void [reset](#) ()  
*Reset the timer.*
- float [get\\_time](#) () const  
*Get elapsed time since last reset.*

## Static Public Member Functions

- static void [sleep](#) (uint32\_t milliseconds)  
*Sleep for a given amount of time.*
- static void [increment\\_system\\_ticks](#) ()  
*Increment the system ticks.*

## 4.9.1 Constructor & Destructor Documentation

### 4.9.1.1 HalTimer()

```
HalTimer::HalTimer ( )
```

Construct a new Hal Timer object.

## 4.9.2 Member Function Documentation

### 4.9.2.1 get\_time()

```
float HalTimer::get_time (
    void ) const
```

Get elapsed time since last reset.

#### Returns

Elapsed time in seconds

#### 4.9.2.2 increment\_system\_ticks()

```
void HalTimer::increment_system_ticks (
    void ) [static]
```

Increment the system ticks.

#### 4.9.2.3 reset()

```
void HalTimer::reset (
    void )
```

Reset the timer.

#### 4.9.2.4 sleep()

```
void HalTimer::sleep (
    uint32_t milliseconds ) [static]
```

Sleep for a given amount of time.

##### Parameters

<i>milliseconds</i>	Time to sleep in milliseconds
---------------------	-------------------------------

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

- inc/hal/[hal\\_timer.hpp](#)
- src/hal/[hal\\_timer.cpp](#)

## 4.10 LineSensors< number\_of\_sensors > Class Template Reference

```
#include <line_sensors.hpp>
```

### Public Member Functions

- [LineSensors](#) (const [AdcConfig](#) &adc\_config)  
*Construct a new Line Sensors object.*
- float [get\\_position](#) ()  
*Gets the line position.*
- void [calibrate\\_white](#) ()  
*Calibrates the line sensors for the white line.*
- void [calibrate\\_black](#) ()  
*Calibrates the line sensors for the black background.*

## 4.10.1 Constructor & Destructor Documentation

### 4.10.1.1 LineSensors()

```
template<uint8_t number_of_sensors>
LineSensors< number_of_sensors >::LineSensors (
    const AdcConfig & adc_config )
```

Construct a new Line Sensors object.

#### Parameters

<i>adc_config</i>	Configuration of the ADC used to read the line sensors
-------------------	--

## 4.10.2 Member Function Documentation

### 4.10.2.1 calibrate\_black()

```
template<uint8_t number_of_sensors>
void LineSensors< number_of_sensors >::calibrate_black
```

Calibrates the line sensors for the black background.

### 4.10.2.2 calibrate\_white()

```
template<uint8_t number_of_sensors>
void LineSensors< number_of_sensors >::calibrate_white
```

Calibrates the line sensors for the white line.

### 4.10.2.3 get\_position()

```
template<uint8_t number_of_sensors>
float LineSensors< number_of_sensors >::get_position
```

Gets the line position.

#### Returns

Position of the line.

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

- [inc/proxy/line\\_sensors.hpp](#)
- [src/proxy/line\\_sensors.cpp](#)

## 4.11 Locomotion Class Reference

```
#include <locomotion.hpp>
```

### Public Member Functions

- [Locomotion](#) (const [MotorConfig](#) &left\_motor\_config, const [MotorConfig](#) &right\_motor\_config, float left\_deadzone=0.0, float right\_deadzone=0.0)  
*Construct a new [Locomotion](#) object.*
- void [set\\_speeds](#) (float linear, float angular)  
*Set the speeds of the motors.*

### Static Public Member Functions

- static float [linear\\_decay](#) (float angular\_error, float dependency)  
*Compute the linear decay of the angular error.*

### 4.11.1 Constructor & Destructor Documentation

#### 4.11.1.1 Locomotion()

```
Locomotion::Locomotion (
    const MotorConfig & left_motor_config,
    const MotorConfig & right_motor_config,
    float left_deadzone = 0.0,
    float right_deadzone = 0.0 )
```

Construct a new [Locomotion](#) object.

#### Parameters

<i>left_motor_config</i>	Configuration of the left motor
<i>right_motor_config</i>	Configuration of the right motor
<i>left_deadzone</i>	Deadzone of the left motor
<i>right_deadzone</i>	Deadzone of the right motor

### 4.11.2 Member Function Documentation

#### 4.11.2.1 linear\_decay()

```
float Locomotion::linear_decay (
    float angular_error,
```

```
float dependency ) [static]
```

Compute the linear decay of the angular error.

#### Parameters

<i>angular_error</i>	Angular error
<i>dependency</i>	Dependency of the linear decay

#### Returns

Linear decay

#### 4.11.2.2 set\_speeds()

```
void Locomotion::set_speeds (
    float linear,
    float angular )
```

Set the speeds of the motors.

#### Parameters

<i>linear</i>	Linear speed
<i>angular</i>	Angular speed

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

- [inc/proxy/locomotion.hpp](#)
- [src/proxy/locomotion.cpp](#)

## 4.12 Motor Class Reference

```
#include <motor.hpp>
```

### Public Member Functions

- [Motor](#) (const [MotorConfig](#) &motor\_config, float deadzone=0.0)  
*Construct a new [Motor](#) object.*
- void [set\\_speed](#) (float speed)  
*Set the speed object.*

#### 4.12.1 Constructor & Destructor Documentation

#### 4.12.1.1 Motor()

```
Motor::Motor (
    const MotorConfig & motor_config,
    float deadzone = 0.0 )
```

Construct a new [Motor](#) object.

##### Parameters

<i>motor_config</i>	Configuration for each pwm of the motor
<i>deadzone</i>	Minimum value of the pwm to start the motor

### 4.12.2 Member Function Documentation

#### 4.12.2.1 set\_speed()

```
void Motor::set_speed (
    float speed )
```

Set the speed object.

##### Parameters

<i>speed</i>	Speed of the motor
--------------	--------------------

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

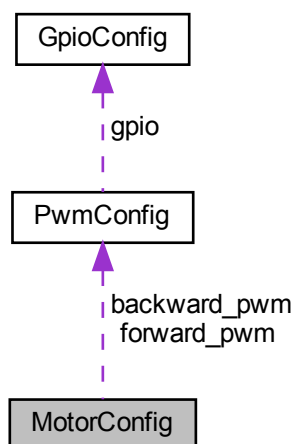
- inc/proxy/[motor.hpp](#)
- src/proxy/[motor.cpp](#)

## 4.13 MotorConfig Struct Reference

```
#include <motor.hpp>
```



Collaboration diagram for MotorConfig:



## Public Attributes

- [PwmConfig forward\\_pwm](#)
- [PwmConfig backward\\_pwm](#)

### 4.13.1 Member Data Documentation

#### 4.13.1.1 backward\_pwm

[PwmConfig](#) MotorConfig::backward\_pwm

#### 4.13.1.2 forward\_pwm

[PwmConfig](#) MotorConfig::forward\_pwm

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

- inc/proxy/[motor.hpp](#)

## 4.14 PidController Class Reference

Implementation of simple PID controller Response =  $K_p(\text{error} + K_i * \text{integral}(\text{error}) + K_d * d/dt(\text{error}))$

```
#include <pid_controller.hpp>
```

### Public Member Functions

- [PidController](#) (float kp, float ki, float kd, float setpoint=0.0, float saturation=-1.0, float max\_integral=-1.0)  
*Construct a new Pid Controller object.*
- void [set\\_setpoint](#) (float setpoint)  
*Set the setpoint object.*
- void [set\\_parameters](#) (float kp, float ki, float kd, float saturation=-1.0, float max\_integral=-1.0)  
*Set the controller parameters.*
- void [reset](#) ()  
*Reset prev\_error and error\_acc objects.*
- float [update](#) (float state)  
*Update PID with new state and return response.*
- float [update](#) (float state, float state\_change)  
*Update PID with new state and return response.*

### 4.14.1 Detailed Description

Implementation of simple PID controller Response =  $K_p(\text{error} + K_i * \text{integral}(\text{error}) + K_d * d/dt(\text{error}))$

### 4.14.2 Constructor & Destructor Documentation

#### 4.14.2.1 PidController()

```
PidController::PidController (
    float kp,
    float ki,
    float kd,
    float setpoint = 0.0,
    float saturation = -1.0,
    float max_integral = -1.0 )
```

Construct a new Pid Controller object.

#### Parameters

<i>kp</i>	Proportional constant
<i>ki</i>	Integrative constant
<i>kd</i>	Derivative constant
<i>setpoint</i>	Desired state
<i>saturation</i>	Maximum response returned by the controller
<i>max_integral</i>	Maximum integrative response

### 4.14.3 Member Function Documentation

#### 4.14.3.1 reset()

```
void PidController::reset (
    void )
```

Reset prev\_error and error\_acc objects.

#### 4.14.3.2 set\_parameters()

```
void PidController::set_parameters (
    float kp,
    float ki,
    float kd,
    float saturation = -1.0,
    float max_integral = -1.0 )
```

Set the controller parameters.

##### Parameters

<i>kp</i>	Proportional constant
<i>ki</i>	Integrative constant
<i>kd</i>	Derivative constant
<i>saturation</i>	Maximum response returned by the controller
<i>max_integral</i>	Maximum integrative response

#### 4.14.3.3 set\_setpoint()

```
void PidController::set_setpoint (
    float setpoint )
```

Set the setpoint object.

##### Parameters

<i>setpoint</i>	Desired state
-----------------	---------------

#### 4.14.3.4 update() [1/2]

```
float PidController::update (
    float state )
```

Update PID with new state and return response.

##### Parameters

<i>state</i>	Current value of the controlled variable
--------------	--

##### Returns

Response

#### 4.14.3.5 update() [2/2]

```
float PidController::update (
    float state,
    float state_change )
```

Update PID with new state and return response.

##### Parameters

<i>state</i>	Current value of the controlled variable
<i>state_change</i>	Derivative of the controlled variable

##### Returns

Response

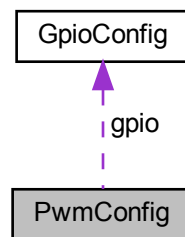
The documentation for this class was generated from the following files:

- [inc/pid\\_controller.hpp](#)
- [src/pid\\_controller.cpp](#)

## 4.15 PwmConfig Struct Reference

```
#include <hal_pwm.hpp>
```

Collaboration diagram for PwmConfig:



## Public Attributes

- [GpioConfig gpio](#)
- `uint32_t` [timer](#)
- `tim_oc_id` [oc\\_id](#)
- `rcc_periph_clken` [rcc\\_clock](#)
- `uint32_t` [period](#)
- `uint32_t` [clock\\_div](#)
- `uint32_t` [prescaler](#)
- `tim_oc_mode` [oc\\_mode](#)

## 4.15.1 Member Data Documentation

### 4.15.1.1 clock\_div

`uint32_t PwmConfig::clock_div`

### 4.15.1.2 gpio

[GpioConfig](#) `PwmConfig::gpio`

### 4.15.1.3 oc\_id

`tim_oc_id PwmConfig::oc_id`

#### 4.15.1.4 oc\_mode

`tim_oc_mode PwmConfig::oc_mode`

#### 4.15.1.5 period

`uint32_t PwmConfig::period`

#### 4.15.1.6 prescaler

`uint32_t PwmConfig::prescaler`

#### 4.15.1.7 rcc\_clock

`rcc_periph_clken PwmConfig::rcc_clock`

#### 4.15.1.8 timer

`uint32_t PwmConfig::timer`

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

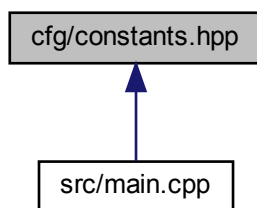
- [inc/hal/hal\\_pwm.hpp](#)

## Chapter 5

# File Documentation

### 5.1 cfg/constants.hpp File Reference

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



#### Variables

- constexpr float [left\\_deadzone](#) = 0.10
- constexpr float [right\\_deadzone](#) = 0.10
- constexpr float [kp](#) = 15.0
- constexpr float [ki](#) = 0.0
- constexpr float [kd](#) = 0.0
- constexpr float [saturation](#) = 100.0
- constexpr float [max\\_integral](#) = 40.0
- constexpr float [filter\\_frequency](#) = 0.5
- constexpr float [linear\\_base\\_speed](#) = 20
- constexpr float [linear\\_decay](#) = 15.0

#### 5.1.1 Variable Documentation

#### 5.1.1.1 filter\_frequency

```
constexpr float filter_frequency = 0.5 [constexpr]
```

#### 5.1.1.2 kd

```
constexpr float kd = 0.0 [constexpr]
```

#### 5.1.1.3 ki

```
constexpr float ki = 0.0 [constexpr]
```

#### 5.1.1.4 kp

```
constexpr float kp = 15.0 [constexpr]
```

#### 5.1.1.5 left\_deadzone

```
constexpr float left_deadzone = 0.10 [constexpr]
```

#### 5.1.1.6 linear\_base\_speed

```
constexpr float linear_base_speed = 20 [constexpr]
```

#### 5.1.1.7 linear\_decay

```
constexpr float linear_decay = 15.0 [constexpr]
```

#### 5.1.1.8 max\_integral

```
constexpr float max_integral = 40.0 [constexpr]
```



### 5.1.1.9 right\_deadzone

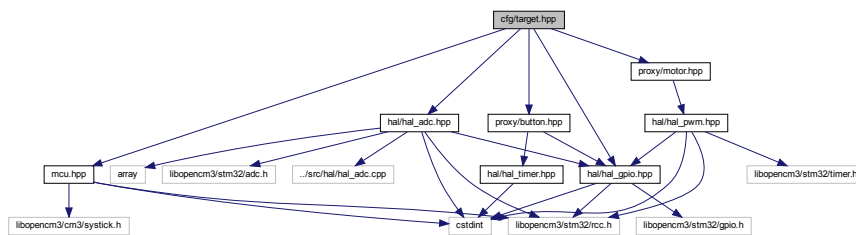
```
constexpr float right_deadzone = 0.10 [constexpr]
```

### 5.1.1.10 saturation

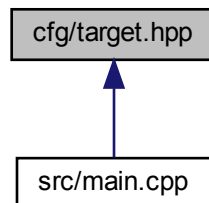
```
constexpr float saturation = 100.0 [constexpr]
```

## 5.2 cfg/target.hpp File Reference

```
#include "mcu.hpp"
#include "hal/hal_gpio.hpp"
#include "hal/hal_adc.hpp"
#include "proxy/motor.hpp"
#include "proxy/button.hpp"
Include dependency graph for target.hpp:
```



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



## Variables

- constexpr [ClockConfig](#) clock\_config
- constexpr [GpioConfig](#) button\_config
- [button\\_pull\\_resistor\\_t](#) button\_pull\_resistor = BUTTON\_PULL\_UP
- constexpr [GpioConfig](#) led\_config
- constexpr [MotorConfig](#) left\_motor\_config
- constexpr [MotorConfig](#) right\_motor\_config
- constexpr [uint8\\_t](#) adc\_num\_channels = 8
- constexpr [uint16\\_t](#) adc\_readings\_per\_channel = 50
- [uint8\\_t](#) adc\_channels [adc\_num\_channels]
- constexpr [AdcConfig](#) line\_sensors\_config

### 5.2.1 Variable Documentation

#### 5.2.1.1 adc\_channels

```
uint8_t adc_channels[adc_num_channels]
```

##### Initial value:

```
= {  
    ADC_CHANNEL0,  
    ADC_CHANNEL1,  
    ADC_CHANNEL2,  
    ADC_CHANNEL3,  
    ADC_CHANNEL4,  
    ADC_CHANNEL5,  
    ADC_CHANNEL6,  
    ADC_CHANNEL7,  
}
```

#### 5.2.1.2 adc\_num\_channels

```
constexpr uint8_t adc_num_channels = 8 [constexpr]
```

#### 5.2.1.3 adc\_readings\_per\_channel

```
constexpr uint16_t adc_readings_per_channel = 50 [constexpr]
```

#### 5.2.1.4 button\_config

```
constexpr GpioConfig button_config [constexpr]
```

**Initial value:**

```
= {  
    .port = GPIOB,  
    .pin = GPIO10,  
    .mode = GPIO_MODE_INPUT,  
    .pull_resistor = GPIO_PUPD_NONE,  
    .rcc_clock = RCC_GPIOB,  
}
```

#### 5.2.1.5 button\_pull\_resistor

```
button\_pull\_resistor\_t button_pull_resistor = BUTTON\_PULL\_UP
```

#### 5.2.1.6 clock\_config

```
constexpr ClockConfig clock_config [constexpr]
```

**Initial value:**

```
= {  
    .clock_scale = &rcc_hse_25mhz_3v3[RCC_CLOCK_3V3_84MHZ],  
    .reload = 84000,  
    .clocksource = STK_CSR_CLKSOURCE_AHB,  
}
```

#### 5.2.1.7 led\_config

```
constexpr GpioConfig led_config [constexpr]
```

**Initial value:**

```
= {  
    .port = GPIOB,  
    .pin = GPIO15,  
    .mode = GPIO_MODE_OUTPUT,  
    .pull_resistor = GPIO_PUPD_NONE,  
    .rcc_clock = RCC_GPIOB,  
    .otype = GPIO_OTYPE_PP,  
    .speed = GPIO_OSPEED_2MHZ,  
}
```

#### 5.2.1.8 left\_motor\_config

```
constexpr MotorConfig left_motor_config [constexpr]
```

### 5.2.1.9 line\_sensors\_config

```
constexpr AdcConfig line_sensors_config [constexpr]
```

#### Initial value:

```
= {  
    .gpio = {  
        .port = GPIOA,  
        .pin = GPIO0 | GPIO1 | GPIO2 | GPIO3 | GPIO4 | GPIO5 | GPIO6 | GPIO7,  
        .mode = GPIO_MODE_ANALOG,  
        .pull_resistor = GPIO_PUPD_NONE,  
        .rcc_clock = RCC_GPIOA,  
    },  
    .adc_number = ADC1,  
    .mode = ADC_CCR_MULTI_INDEPENDENT,  
    .rcc_clock = RCC_ADC1,  
    .rcc_reset = RST_ADC,  
    .prescaler = ADC_CCR_ADCPRE_BY4,  
    .resolution = ADC_CR1_RES_12BIT,  
    .channels = adc\_channels,  
    .sample_time = ADC_SMPR_SMP_56CYC,  
}
```

### 5.2.1.10 right\_motor\_config

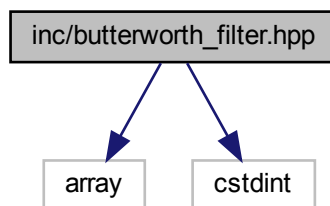
```
constexpr MotorConfig right_motor_config [constexpr]
```

## 5.3 inc/butterworth\_filter.hpp File Reference

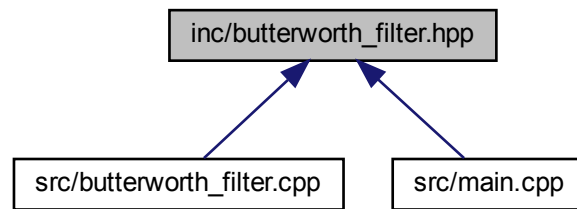
```
#include <array>
```

```
#include <cstdint>
```

Include dependency graph for butterworth\_filter.hpp:



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



## Classes

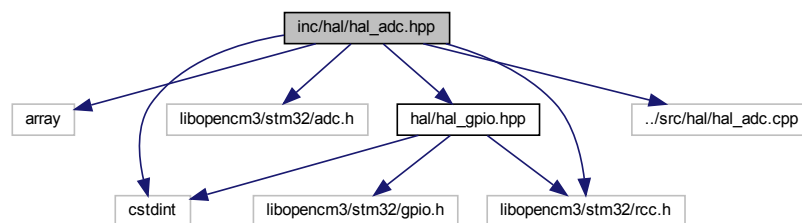
- class [ButterworthFilter](#)

Implementation of Butterworth second order low-pass filter A generic digital filter follows the relation  $a_0 * y[k] = \sum(b_i * x[k - i]) - \sum(a_j * y[k - j])$  Where  $x[k]$  - measurement at instant  $k$   $y[k]$  - filtered signal at instant  $k$  The Butterworth filter have the special property of being a maximally flat magnitude filter, in other words, is the best filter that doesn't present distortions around the cutoff frequency The formula for the continuous coefficients of the Butterworth filter is available here: [https://en.wikipedia.org/wiki/Butterworth\\_filter](https://en.wikipedia.org/wiki/Butterworth_filter) The discrete version were computed with the Tustin method: [https://en.wikipedia.org/wiki/Bilinear\\_transform](https://en.wikipedia.org/wiki/Bilinear_transform).

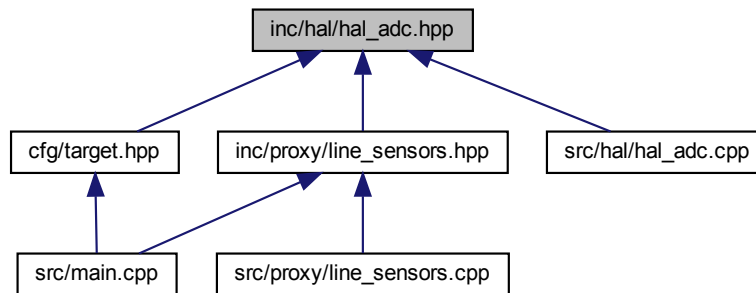
## 5.4 inc/hal/hal\_adc.hpp File Reference

```
#include <array>
#include <cstdint>
#include <libopencm3/stm32/adc.h>
#include <libopencm3/stm32/rcc.h>
#include "hal/hal_gpio.hpp"
#include "../src/hal/hal_adc.cpp"
```

Include dependency graph for hal\_adc.hpp:



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



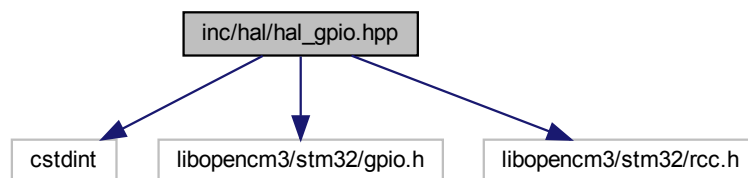
## Classes

- struct [AdcConfig](#)
- class [HalAdc](#)< number\_of\_channels >

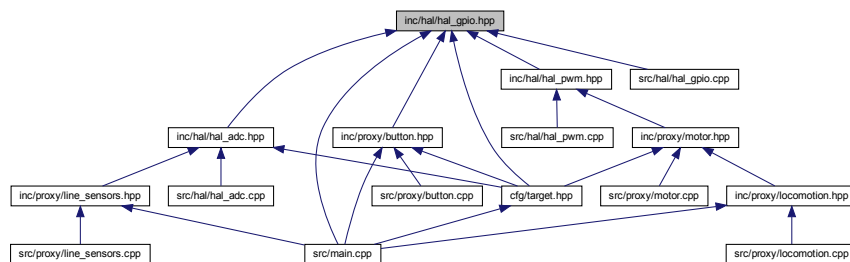
## 5.5 inc/hal/hal\_gpio.hpp File Reference

```
#include <cstdint>
#include <libopencm3/stm32/gpio.h>
#include <libopencm3/stm32/rcc.h>
```

Include dependency graph for `hal_gpio.hpp`:



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



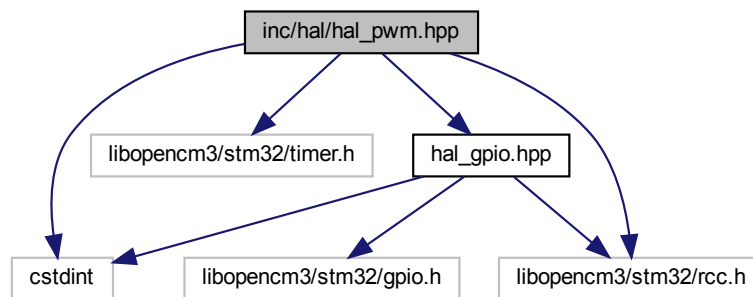
## Classes

- struct [GpioConfig](#)
- class [HalGpio](#)

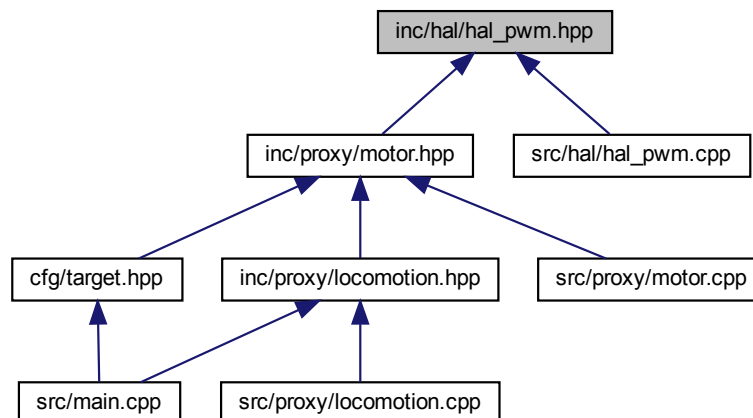
## 5.6 inc/hal/hal\_pwm.hpp File Reference

```
#include <cstdint>
#include <libopencm3/stm32/timer.h>
#include <libopencm3/stm32/rcc.h>
#include "hal_gpio.hpp"
```

Include dependency graph for hal\_pwm.hpp:



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



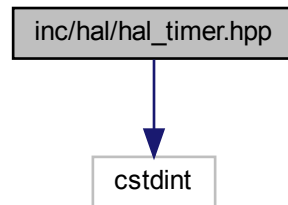
## Classes

- struct [PwmConfig](#)
- class [HalPwm](#)

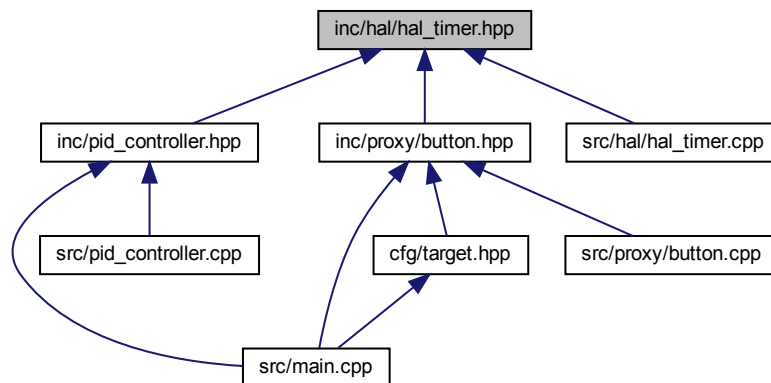
## 5.7 inc/hal/hal\_timer.hpp File Reference

```
#include <stdint>
```

Include dependency graph for hal\_timer.hpp:



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



### Classes

- class [HalTimer](#)

## 5.8 inc/mcu.hpp File Reference

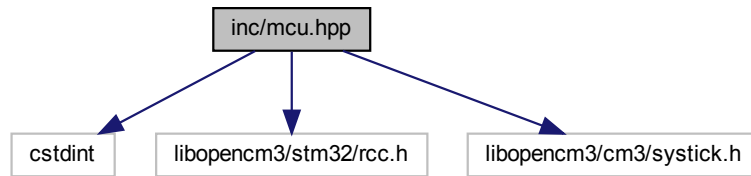
```
#include <stdint>
```

```
#include <libopencm3/stm32/rcc.h>
```

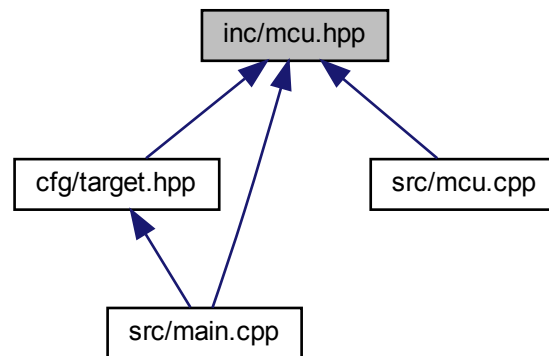


```
#include <libopencm3/cm3/systick.h>
```

Include dependency graph for mcu.hpp:



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



## Classes

- struct [ClockConfig](#)

## Functions

- void [mcu\\_init](#) (const [ClockConfig](#) &[clock\\_config](#))  
*Initializes MCU and some peripherals.*

### 5.8.1 Function Documentation

#### 5.8.1.1 mcu\_init()

```
void mcu_init (
    const ClockConfig & clock_config )
```

Initializes MCU and some peripherals.

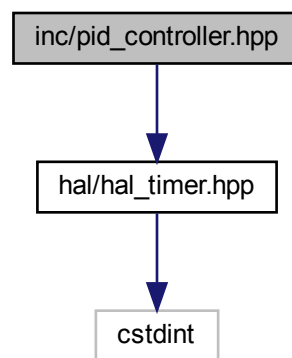
## Parameters

<code>clock_config</code>	Configuration of the clock
---------------------------	----------------------------

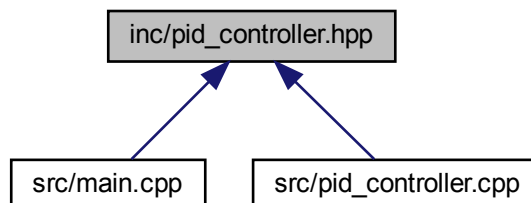
## 5.9 inc/pid\_controller.hpp File Reference

```
#include <hal/hal_timer.hpp>
```

Include dependency graph for pid\_controller.hpp:



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



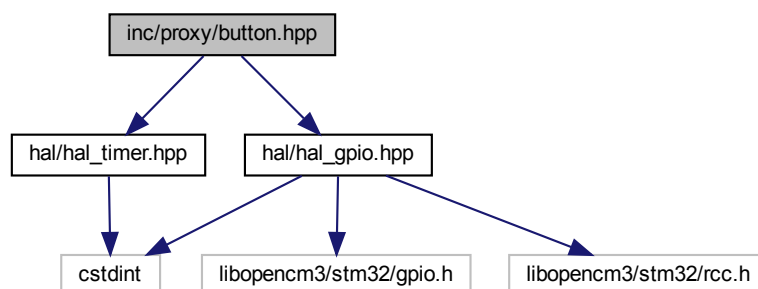
## Classes

- class [PidController](#)

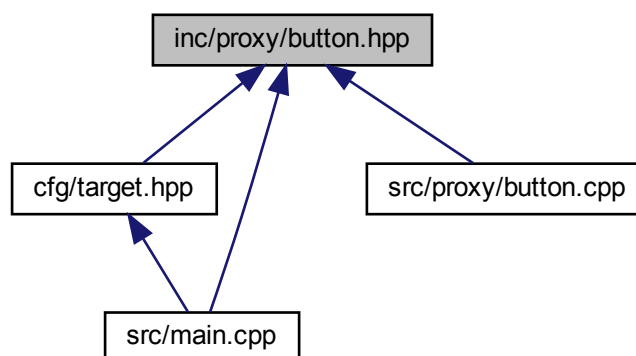
*Implementation of simple PID controller Response =  $K_p(\text{error} + K_i * \text{integral}(\text{error}) + K_d * d/dt(\text{error}))$*

## 5.10 inc/proxy/button.hpp File Reference

```
#include "hal/hal_gpio.hpp"
#include "hal/hal_timer.hpp"
Include dependency graph for button.hpp:
```



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



### Classes

- class [Button](#)

### Enumerations

- enum [button\\_status\\_t](#) { [BUTTON\\_NO\\_PRESS](#) , [BUTTON\\_SHORT\\_PRESS](#) , [BUTTON\\_LONG\\_PRESS](#) , [BUTTON\\_EXTRA\\_LONG\\_PRESS](#) }  
*Button status type.*
- enum [button\\_pull\\_resistor\\_t](#) { [BUTTON\\_PULL\\_UP](#) , [BUTTON\\_PULL\\_DOWN](#) }  
*Type of pull resistor configuration.*

## 5.10.1 Enumeration Type Documentation

### 5.10.1.1 button\_pull\_resistor\_t

enum `button_pull_resistor_t`

Type of pull resistor configuration.

Enumerator

BUTTON_PULL_UP	
BUTTON_PULL_DOWN	

### 5.10.1.2 button\_status\_t

enum `button_status_t`

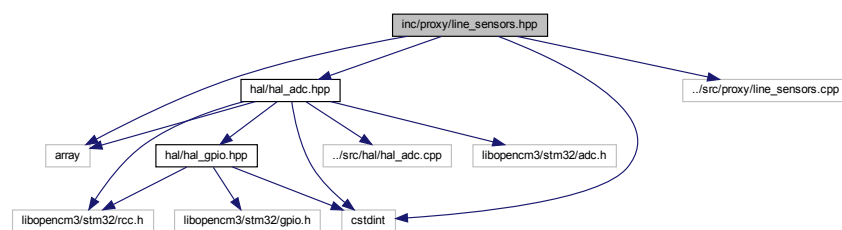
`Button` status type.

Enumerator

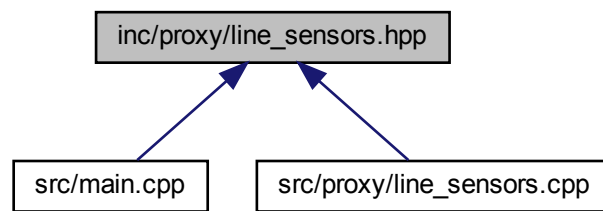
BUTTON_NO_PRESS	
BUTTON_SHORT_PRESS	
BUTTON_LONG_PRESS	
BUTTON_EXTRA_LONG_PRESS	

## 5.11 inc/proxy/line\_sensors.hpp File Reference

```
#include <array>
#include <cstdint>
#include "hal/hal_adc.hpp"
#include "../src/proxy/line_sensors.cpp"
Include dependency graph for line_sensors.hpp:
```



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



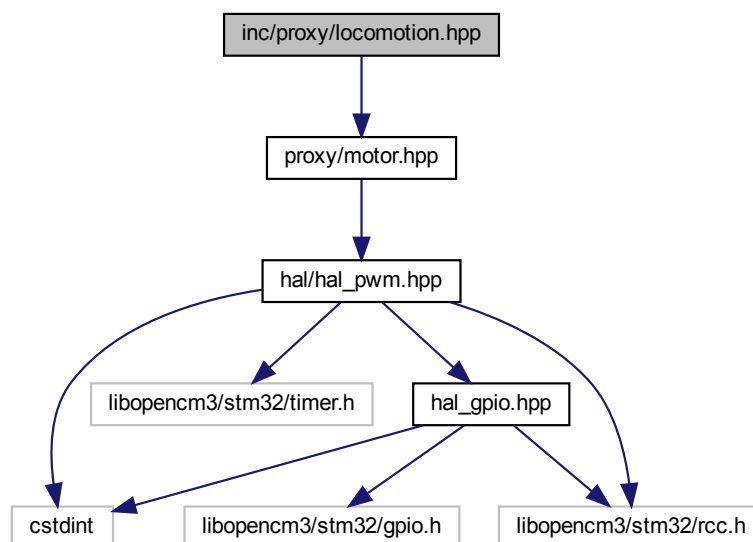
## Classes

- class [LineSensors](#)< number\_of\_sensors >

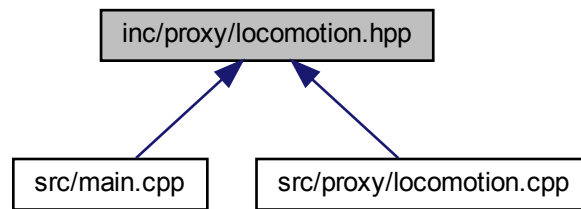
## 5.12 inc/proxy/locomotion.hpp File Reference

```
#include "proxy/motor.hpp"
```

Include dependency graph for locomotion.hpp:



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



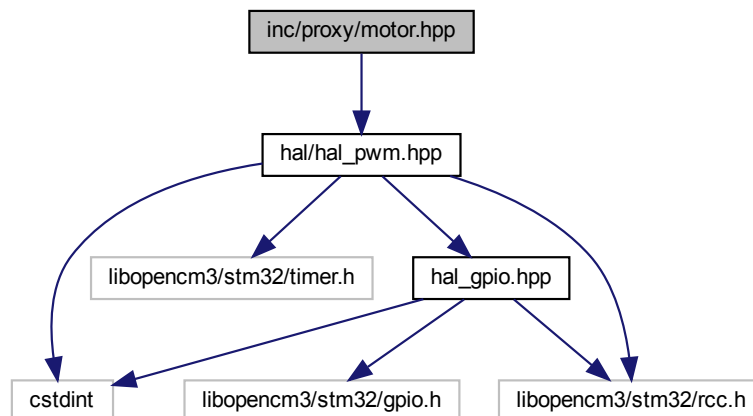
## Classes

- class [Locomotion](#)

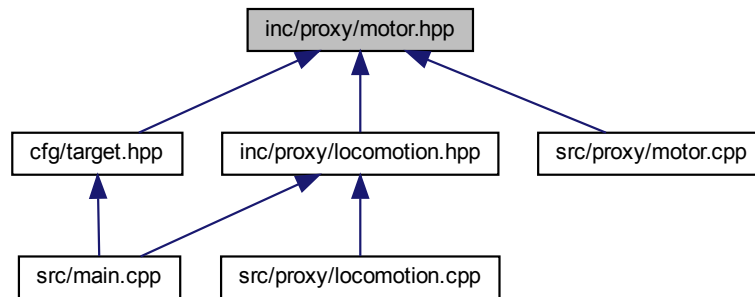
## 5.13 inc/proxy/motor.hpp File Reference

```
#include "hal/hal_pwm.hpp"
```

Include dependency graph for motor.hpp:



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



## Classes

- struct [MotorConfig](#)
- class [Motor](#)

## Variables

- constexpr float [max\\_motors\\_speed](#) = 100.0
- constexpr float [min\\_motors\\_speed](#) = -[max\\_motors\\_speed](#)

### 5.13.1 Variable Documentation

#### 5.13.1.1 max\_motors\_speed

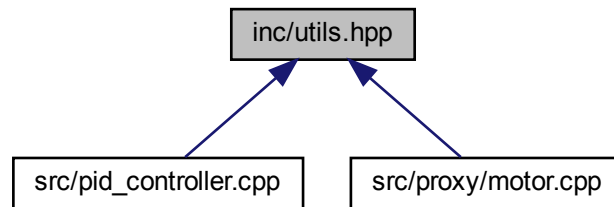
```
constexpr float max_motors_speed = 100.0 [constexpr]
```

#### 5.13.1.2 min\_motors\_speed

```
constexpr float min_motors_speed = -max_motors_speed [constexpr]
```

## 5.14 inc/utls.hpp File Reference

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



### Macros

- `#define constrain(value, min, max) (value < min ? min : (value > max ? max : value))`
- `#define map(value, from_min, from_max, to_min, to_max) (to_min + (to_max - to_min) * (value - from_min) / (from_max - from_min))`

### 5.14.1 Macro Definition Documentation

#### 5.14.1.1 constrain

```
#define constrain(
    value,
    min,
    max ) (value < min ? min : (value > max ? max : value))
```

#### 5.14.1.2 map

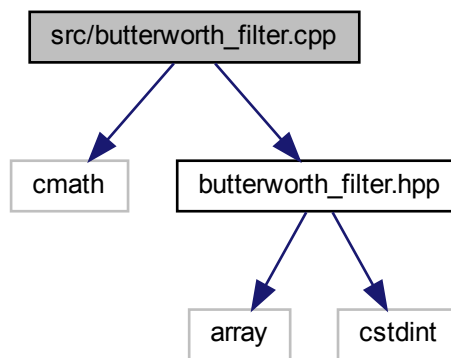
```
#define map(
    value,
    from_min,
    from_max,
    to_min,
    to_max ) (to_min + (to_max - to_min) * (value - from_min) / (from_max - from_min))
```



## 5.15 README.md File Reference

## 5.16 src/butterworth\_filter.cpp File Reference

```
#include <cmath>
#include "butterworth_filter.hpp"
Include dependency graph for butterworth_filter.cpp:
```



### Variables

- `constexpr float sqrt2 = 1.41421356237309504880168872420969807856967187537694807317667973799`

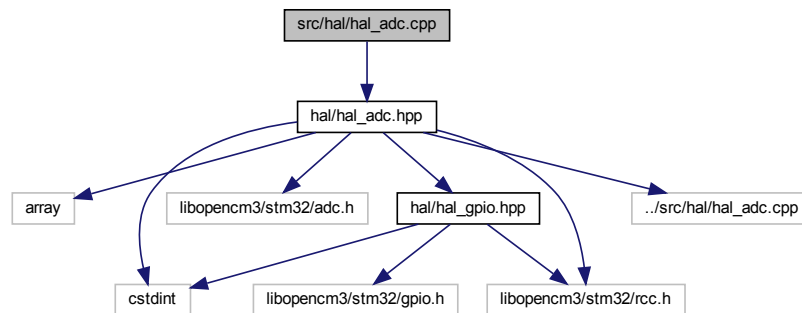
### 5.16.1 Variable Documentation

#### 5.16.1.1 sqrt2

```
constexpr float sqrt2 = 1.41421356237309504880168872420969807856967187537694807317667973799
[constexpr]
```

## 5.17 src/hal/hal\_adc.cpp File Reference

```
#include "hal/hal_adc.hpp"
Include dependency graph for hal_adc.cpp:
```



### Macros

- `#define __HAL_ADC_CPP__`

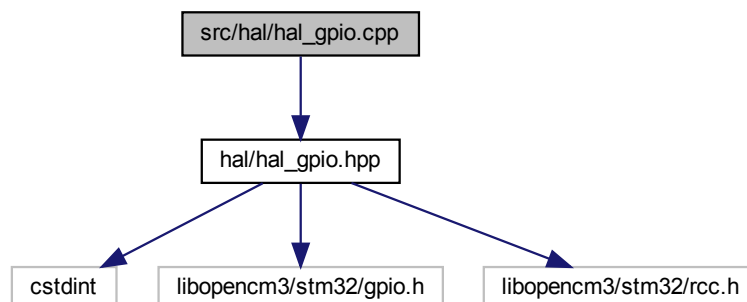
### 5.17.1 Macro Definition Documentation

#### 5.17.1.1 \_\_HAL\_ADC\_CPP\_\_

```
#define __HAL_ADC_CPP__
```

## 5.18 src/hal/hal\_gpio.cpp File Reference

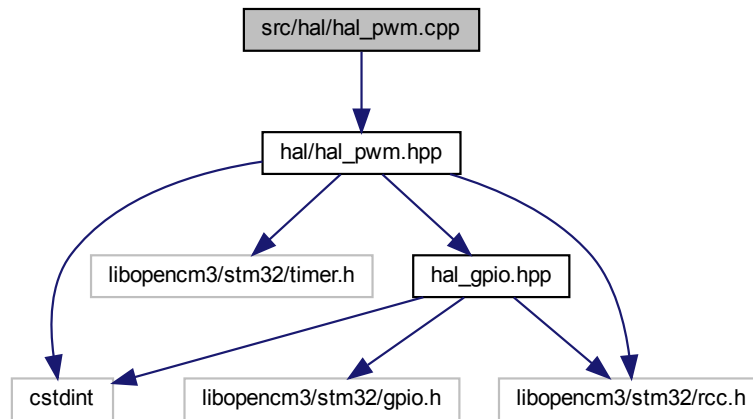
```
#include "hal/hal_gpio.hpp"
Include dependency graph for hal_gpio.cpp:
```



## 5.19 src/hal/hal\_pwm.cpp File Reference

```
#include "hal/hal_pwm.hpp"
```

Include dependency graph for hal\_pwm.cpp:

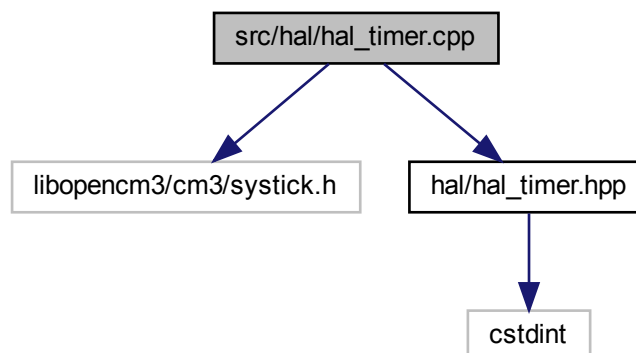


## 5.20 src/hal/hal\_timer.cpp File Reference

```
#include <libopencm3/cm3/systick.h>
```

```
#include "hal/hal_timer.hpp"
```

Include dependency graph for hal\_timer.cpp:



### Functions

- void [sys\\_tick\\_handler](#) (void)



### 5.21.1.1 main()

```
int main (  
    void )
```

## 5.21.2 Variable Documentation

### 5.21.2.1 angular\_command

```
float angular_command = 0
```

### 5.21.2.2 angular\_position

```
float angular_position = 0
```

### 5.21.2.3 line\_measure

```
float line_measure = 0
```

### 5.21.2.4 linear\_command

```
float linear_command = 0
```

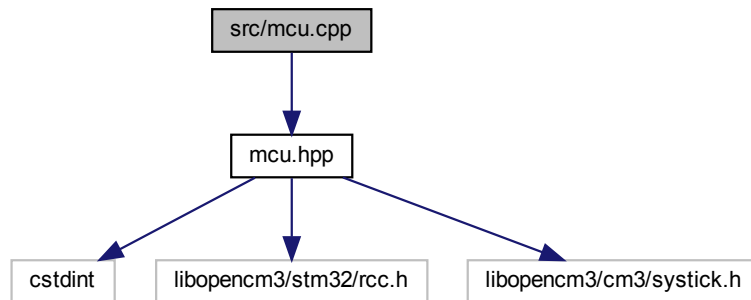
### 5.21.2.5 stopped

```
bool stopped = true
```

## 5.22 src/mcu.cpp File Reference

```
#include "mcu.hpp"
```

Include dependency graph for mcu.cpp:



### Functions

- void `mcu_init` (const `ClockConfig` & `clock_config`)  
*Initializes MCU and some peripherals.*

### 5.22.1 Function Documentation

#### 5.22.1.1 mcu\_init()

```
void mcu_init (
    const ClockConfig & clock_config )
```

Initializes MCU and some peripherals.

#### Parameters

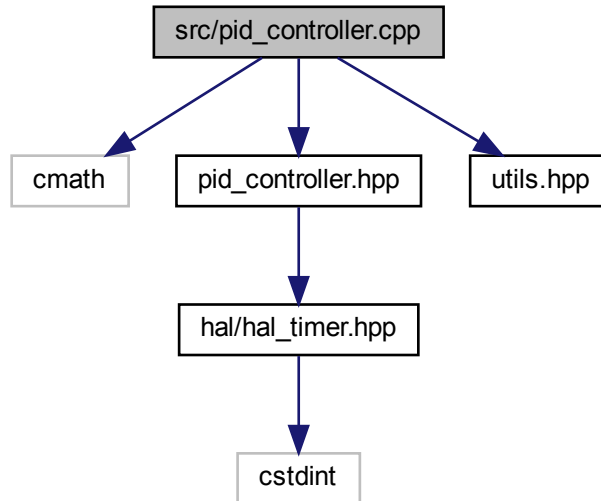
<code>clock_config</code>	Configuration of the clock
---------------------------	----------------------------

## 5.23 src/pid\_controller.cpp File Reference

```
#include <cmath>
#include "pid_controller.hpp"
```

```
#include "utils.hpp"
```

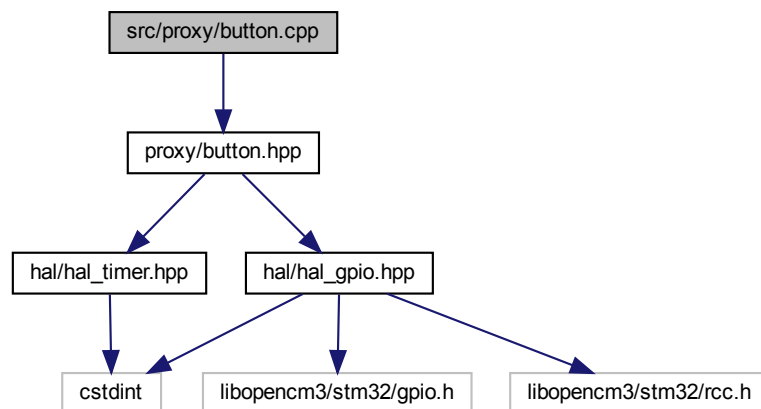
Include dependency graph for pid\_controller.cpp:



## 5.24 src/proxy/button.cpp File Reference

```
#include "proxy/button.hpp"
```

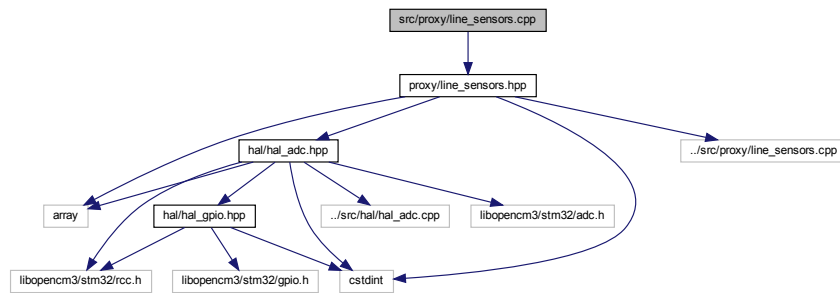
Include dependency graph for button.cpp:



## 5.25 src/proxy/line\_sensors.cpp File Reference

```
#include "proxy/line_sensors.hpp"
```

Include dependency graph for line\_sensors.cpp:



## Macros

- `#define __LINE_SENSORS_CPP__`

## Variables

- `constexpr uint32_t default_white_value = 3850`
- `constexpr uint32_t default_black_value = 4000`

### 5.25.1 Macro Definition Documentation

#### 5.25.1.1 \_\_LINE\_SENSORS\_CPP\_\_

```
#define __LINE_SENSORS_CPP__
```

### 5.25.2 Variable Documentation

#### 5.25.2.1 default\_black\_value

```
constexpr uint32_t default_black_value = 4000 [constexpr]
```

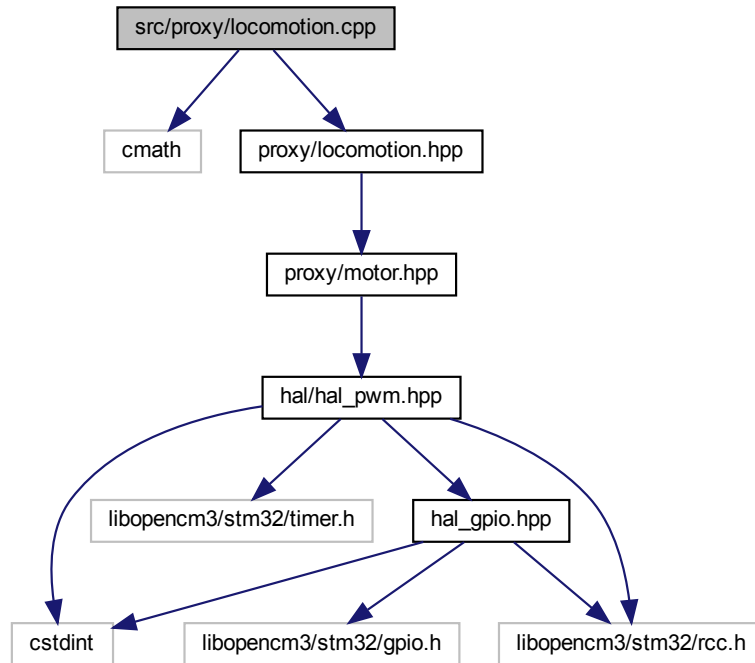
#### 5.25.2.2 default\_white\_value

```
constexpr uint32_t default_white_value = 3850 [constexpr]
```



## 5.26 src/proxy/locomotion.cpp File Reference

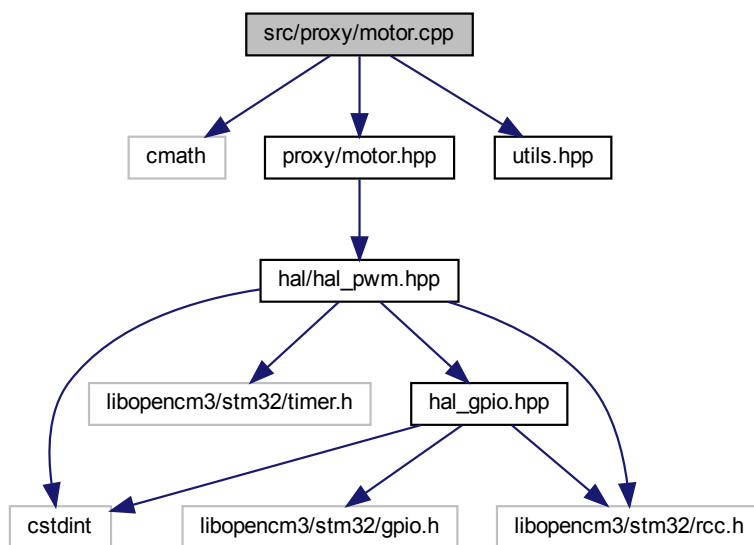
```
#include <cmath>
#include "proxy/locomotion.hpp"
Include dependency graph for locomotion.cpp:
```



## 5.27 src/proxy/motor.cpp File Reference

```
#include <cmath>
#include "proxy/motor.hpp"
#include "utils.hpp"
```

Include dependency graph for motor.cpp:



# Index

- `__HAL_ADC_CPP__`
    - `hal_adc.cpp`, [50](#)
  - `__LINE_SENSORS_CPP__`
    - `line_sensors.cpp`, [56](#)
- `adc_channels`
  - `target.hpp`, [34](#)
- `adc_num_channels`
  - `target.hpp`, [34](#)
- `adc_number`
  - `AdcConfig`, [7](#)
- `adc_readings_per_channel`
  - `target.hpp`, [34](#)
- `AdcConfig`, [7](#)
  - `adc_number`, [7](#)
  - `channels`, [8](#)
  - `gpio`, [8](#)
  - `mode`, [8](#)
  - `prescaler`, [8](#)
  - `rcc_clock`, [8](#)
  - `rcc_reset`, [8](#)
  - `resolution`, [8](#)
  - `sample_time`, [8](#)
- `alt_func_num`
  - `GpioConfig`, [13](#)
- `angular_command`
  - `main.cpp`, [53](#)
- `angular_position`
  - `main.cpp`, [53](#)
- `backward_pwm`
  - `MotorConfig`, [25](#)
- `butterworth_filter.cpp`
  - `sqrt2`, [49](#)
- `ButterworthFilter`, [9](#)
  - `ButterworthFilter`, [9](#)
  - `update`, [10](#)
- `Button`, [10](#)
  - `Button`, [10](#)
  - `get_status`, [12](#)
- `button.hpp`
  - `BUTTON_EXTRA_LONG_PRESS`, [44](#)
  - `BUTTON_LONG_PRESS`, [44](#)
  - `BUTTON_NO_PRESS`, [44](#)
  - `BUTTON_PULL_DOWN`, [44](#)
  - `button_pull_resistor_t`, [44](#)
  - `BUTTON_PULL_UP`, [44](#)
  - `BUTTON_SHORT_PRESS`, [44](#)
  - `button_status_t`, [44](#)
- `button_config`
  - `target.hpp`, [34](#)
- `BUTTON_EXTRA_LONG_PRESS`
  - `button.hpp`, [44](#)
- `BUTTON_LONG_PRESS`
  - `button.hpp`, [44](#)
- `BUTTON_NO_PRESS`
  - `button.hpp`, [44](#)
- `BUTTON_PULL_DOWN`
  - `button.hpp`, [44](#)
- `button_pull_resistor`
  - `target.hpp`, [35](#)
- `button_pull_resistor_t`
  - `button.hpp`, [44](#)
- `BUTTON_PULL_UP`
  - `button.hpp`, [44](#)
- `BUTTON_SHORT_PRESS`
  - `button.hpp`, [44](#)
- `button_status_t`
  - `button.hpp`, [44](#)
- `calibrate_black`
  - `LineSensors< number_of_sensors >`, [21](#)
- `calibrate_white`
  - `LineSensors< number_of_sensors >`, [21](#)
- `cfg/constants.hpp`, [31](#)
- `cfg/target.hpp`, [33](#)
- `channels`
  - `AdcConfig`, [8](#)
- `clock_config`
  - `target.hpp`, [35](#)
- `clock_div`
  - `PwmConfig`, [29](#)
- `clock_scale`
  - `ClockConfig`, [12](#)
- `ClockConfig`, [12](#)
  - `clock_scale`, [12](#)
  - `clocksource`, [12](#)
  - `reload`, [13](#)
- `clocksource`
  - `ClockConfig`, [12](#)
- `constants.hpp`
  - `filter_frequency`, [31](#)
  - `kd`, [32](#)
  - `ki`, [32](#)
  - `kp`, [32](#)
  - `left_deadzone`, [32](#)
  - `linear_base_speed`, [32](#)
  - `linear_decay`, [32](#)
  - `max_integral`, [32](#)
  - `right_deadzone`, [32](#)

- saturation, 33
- constrain
  - utils.hpp, 48
- default\_black\_value
  - line\_sensors.cpp, 56
- default\_white\_value
  - line\_sensors.cpp, 56
- filter\_frequency
  - constants.hpp, 31
- forward\_pwm
  - MotorConfig, 25
- get\_adc\_reading
  - HalAdc< number\_of\_channels >, 15
- get\_position
  - LineSensors< number\_of\_sensors >, 21
- get\_status
  - Button, 12
- get\_time
  - HalTimer, 19
- gpio
  - AdcConfig, 8
  - PwmConfig, 29
- GpioConfig, 13
  - alt\_func\_num, 13
  - mode, 13
  - otype, 13
  - pin, 14
  - port, 14
  - pull\_resistor, 14
  - rcc\_clock, 14
  - speed, 14
- hal\_adc.cpp
  - \_\_HAL\_ADC\_CPP\_\_, 50
- hal\_timer.cpp
  - sys\_tick\_handler, 52
- HalAdc
  - HalAdc< number\_of\_channels >, 15
- HalAdc< number\_of\_channels >, 14
  - get\_adc\_reading, 15
  - HalAdc, 15
  - update\_reading, 15
- HalGpio, 16
  - HalGpio, 16
  - read, 17
  - toggle, 17
  - write, 17
- HalPwm, 17
  - HalPwm, 18
  - set\_compare, 18
- HalTimer, 18
  - get\_time, 19
  - HalTimer, 19
  - increment\_system\_ticks, 19
  - reset, 20
  - sleep, 20
- inc/butterworth\_filter.hpp, 36
- inc/hal/hal\_adc.hpp, 37
- inc/hal/hal\_gpio.hpp, 38
- inc/hal/hal\_pwm.hpp, 39
- inc/hal/hal\_timer.hpp, 40
- inc/mcu.hpp, 40
- inc/pid\_controller.hpp, 42
- inc/proxy/button.hpp, 43
- inc/proxy/line\_sensors.hpp, 44
- inc/proxy/locomotion.hpp, 45
- inc/proxy/motor.hpp, 46
- inc/utils.hpp, 48
- increment\_system\_ticks
  - HalTimer, 19
- kd
  - constants.hpp, 32
- ki
  - constants.hpp, 32
- kp
  - constants.hpp, 32
- led\_config
  - target.hpp, 35
- left\_deadzone
  - constants.hpp, 32
- left\_motor\_config
  - target.hpp, 35
- line\_measure
  - main.cpp, 53
- line\_sensors.cpp
  - \_\_LINE\_SENSORS\_CPP\_\_, 56
  - default\_black\_value, 56
  - default\_white\_value, 56
- line\_sensors\_config
  - target.hpp, 35
- linear\_base\_speed
  - constants.hpp, 32
- linear\_command
  - main.cpp, 53
- linear\_decay
  - constants.hpp, 32
  - Locomotion, 22
- LineSensors
  - LineSensors< number\_of\_sensors >, 21
- LineSensors< number\_of\_sensors >, 20
  - calibrate\_black, 21
  - calibrate\_white, 21
  - get\_position, 21
  - LineSensors, 21
- Locomotion, 22
  - linear\_decay, 22
  - Locomotion, 22
  - set\_speeds, 23
- main
  - main.cpp, 52
- main.cpp
  - angular\_command, 53

- angular\_position, 53
- line\_measure, 53
- linear\_command, 53
- main, 52
- stopped, 53
- map
  - utils.hpp, 48
- max\_integral
  - constants.hpp, 32
- max\_motors\_speed
  - motor.hpp, 47
- mcu.cpp
  - mcu\_init, 54
- mcu.hpp
  - mcu\_init, 41
- mcu\_init
  - mcu.cpp, 54
  - mcu.hpp, 41
- min\_motors\_speed
  - motor.hpp, 47
- mode
  - AdcConfig, 8
  - GpioConfig, 13
- Motor, 23
  - Motor, 23
  - set\_speed, 24
- motor.hpp
  - max\_motors\_speed, 47
  - min\_motors\_speed, 47
- MotorConfig, 24
  - backward\_pwm, 25
  - forward\_pwm, 25
- oc\_id
  - PwmConfig, 29
- oc\_mode
  - PwmConfig, 29
- otype
  - GpioConfig, 13
- period
  - PwmConfig, 30
- PidController, 26
  - PidController, 26
  - reset, 27
  - set\_parameters, 27
  - set\_setpoint, 27
  - update, 27, 28
- pin
  - GpioConfig, 14
- port
  - GpioConfig, 14
- prescaler
  - AdcConfig, 8
  - PwmConfig, 30
- pull\_resistor
  - GpioConfig, 14
- PwmConfig, 28
  - clock\_div, 29
  - gpio, 29
  - oc\_id, 29
  - oc\_mode, 29
  - period, 30
  - prescaler, 30
  - rcc\_clock, 30
  - timer, 30
- rcc\_clock
  - AdcConfig, 8
  - GpioConfig, 14
  - PwmConfig, 30
- rcc\_reset
  - AdcConfig, 8
- read
  - HalGpio, 17
- README.md, 49
- reload
  - ClockConfig, 13
- reset
  - HalTimer, 20
  - PidController, 27
- resolution
  - AdcConfig, 8
- right\_deadzone
  - constants.hpp, 32
- right\_motor\_config
  - target.hpp, 36
- sample\_time
  - AdcConfig, 8
- saturation
  - constants.hpp, 33
- set\_compare
  - HalPwm, 18
- set\_parameters
  - PidController, 27
- set\_setpoint
  - PidController, 27
- set\_speed
  - Motor, 24
- set\_speeds
  - Locomotion, 23
- sleep
  - HalTimer, 20
- speed
  - GpioConfig, 14
- src/butterworth\_filter.cpp, 49
- src/hal/hal\_adc.cpp, 50
- src/hal/hal\_gpio.cpp, 50
- src/hal/hal\_pwm.cpp, 51
- src/hal/hal\_timer.cpp, 51
- src/main.cpp, 52
- src/mcu.cpp, 54
- src/pid\_controller.cpp, 54
- src/proxy/button.cpp, 55
- src/proxy/line\_sensors.cpp, 55
- src/proxy/locomotion.cpp, 57
- src/proxy/motor.cpp, 57

- sqrt2
  - butterworth\_filter.cpp, [49](#)
- stopped
  - main.cpp, [53](#)
- sys\_tick\_handler
  - hal\_timer.cpp, [52](#)
- target.hpp
  - adc\_channels, [34](#)
  - adc\_num\_channels, [34](#)
  - adc\_readings\_per\_channel, [34](#)
  - button\_config, [34](#)
  - button\_pull\_resistor, [35](#)
  - clock\_config, [35](#)
  - led\_config, [35](#)
  - left\_motor\_config, [35](#)
  - line\_sensors\_config, [35](#)
  - right\_motor\_config, [36](#)
- timer
  - PwmConfig, [30](#)
- toggle
  - HalGpio, [17](#)
- update
  - ButterworthFilter, [10](#)
  - PidController, [27](#), [28](#)
- update\_reading
  - HalAdc< number\_of\_channels >, [15](#)
- utils.hpp
  - constrain, [48](#)
  - map, [48](#)
- write
  - HalGpio, [17](#)