Prova Prática - Laboratório de Processadores

Generated by Doxygen 1.9.1

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

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 hal Namespace Reference

Classes

- class Adc
- class Clock
- class Gpio
- class Pwm
- class Timer

5.2 proxy Namespace Reference

Classes

- class Button
- class LineSensors
- class Locomotion
- class Motor

Variables

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

5.2.1 Variable Documentation

5.2.1.1 default_black_value

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

5.2.1.2 default_white_value

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

Chapter 6

Class Documentation

6.1 hal::Adc< number_of_channels > Class Template Reference

```
#include <adc.hpp>
```

Classes

- struct Config

Public Member Functions

- Adc (const Config &adc_config)

  Construct a new Adc object.
- void update_reading ()

  Update the ADC reading.
- uint32_t get_reading (uint8_t channel) const

  Get the reading of the ADC.

6.1.1 Constructor & Destructor Documentation

6.1.1.1 Adc()

```
template<uint8_t number_of_channels>
hal::Adc< number_of_channels >::Adc (
            const Config & adc_config )
```

Construct a new Adc object.

Parameters

adc_config	Configuration of the ADC

6.1.2 Member Function Documentation

6.1.2.1 get_reading()

```
template<uint8_t number_of_channels>
uint32_t hal::Adc< number_of_channels >::get_reading (
            uint8_t channel ) const
```

Get the reading of the ADC.

Parameters

channel	Channel of the ADC

Returns

uint32_t Reading of the ADC channel

6.1.2.2 update_reading()

```
template<uint8_t number_of_channels>
void hal::Adc< number_of_channels >::update_reading (
            void )
```

Update the ADC reading.

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

- inc/hal/adc.hpp
- src/hal/adc.cpp

6.2 ButterworthFilter Class Reference

Implementation of Butterworth second order low-pass filter A generic digital filter follows the relation a0 ∗ y[k] = sum(bi ∗ x[k - i]) - sum(aj ∗ 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 continuos coefficients of the Butterworth filter is available here: https://en.wikipedia.org/wiki/Butterworth_filter The discrete version were computed with the Tustin method: 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.

6.2.1 Detailed Description

Implementation of Butterworth second order low-pass filter A generic digital filter follows the relation a0 ∗ y[k] = sum(bi ∗ x[k - i]) - sum(aj ∗ 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 continuos coefficients of the Butterworth filter is available here: `https://en.wikipedia.org/wiki/Butterworth_filter` The discrete version were computed with the Tustin method: `https://en.wikipedia.org/wiki/↩ Bilinear_transform`.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ButterworthFilter()

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

Construct a new Butterworth Second Order filter object.

Parameters

cutoff_frequency	Low-pass cutoff frequency in Hz
sampling_frequency	Sampling frequency in Hz.

6.2.3 Member Function Documentation

6.2.3.1 update()

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

Produces a new value from measured data.

Parameters

x0	Last measure

Returns

Filtered value

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

- inc/butterworth_filter.hpp
- src/butterworth_filter.cpp

6.3 proxy::Button Class Reference

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

Public Types

- enum status_t { NO_PRESS , SHORT_PRESS , LONG_PRESS , EXTRA_LONG_PRESS }

  Button status type.
- enum pull_resistor_t { PULL_UP , PULL_DOWN }

  Type of pull resistor configuration.

Public Member Functions

- Button (const hal::Gpio::Config &gpio_config, pull_resistor_t pull_resistor)

  Construct a new Button object.
- status_t get_status ()

  Provides the status of the chosen button.

6.3.1 Member Enumeration Documentation

6.3.1.1 pull_resistor_t

```
enum proxy::Button::pull_resistor_t
```

Type of pull resistor configuration.

Enumerator

PULL_UP	
PULL_DOWN	

6.3.1.2 status_t

```
enum proxy::Button::status_t
```

Button status type.

Enumerator

NO_PRESS	
SHORT_PRESS	
LONG_PRESS	
EXTRA_LONG_PRESS	

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Button()

```
proxy::Button::Button (
            const hal::Gpio::Config & gpio_config,
            pull_resistor_t pull_resistor )
```

Construct a new Button object.

Parameters

pio_config	Configuration of the button GPIO port
pull_resistor	Type of pull resistor configuration

6.3.3 Member Function Documentation

6.3.3.1 get_status()

```
Button::status_t proxy::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

6.4 hal::Clock Class Reference

```
#include <clock.hpp>
```

Classes

- struct Config

Public Member Functions

- Clock ()=delete

Static Public Member Functions

- static void init (const Config &clock_config)

  Configure and initializes system clock.

6.4.1 Constructor & Destructor Documentation

6.4.1.1 Clock()

```
hal::Clock::Clock ( )  [delete]
```

6.4.2 Member Function Documentation

6.4.2.1 init()

```
void hal::Clock::init (
            const Config & clock_config )  [static]
```

Configure and initializes system clock.

Parameters

clock_config	Configuration of the clock

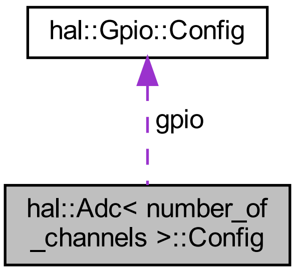
The documentation for this class was generated from the following files:

- inc/hal/clock.hpp
- src/hal/clock.cpp

6.5 hal::Adc< number_of_channels >::Config Struct Reference

```
#include <adc.hpp>
```

Collaboration diagram for hal::Adc< number_of_channels >::Config:



Public Attributes

- Gpio::Config 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

6.5.1 Member Data Documentation

6.5.1.1 adc_number

```
template<uint8_t number_of_channels>
uint32_t hal::Adc< number_of_channels >::Config::adc_number
```

6.5.1.2 channels

```
template<uint8_t number_of_channels>
uint8_t∗ hal::Adc< number_of_channels >::Config::channels
```

6.5.1.3 gpio

```
template<uint8_t number_of_channels>
Gpio::Config hal::Adc< number_of_channels >::Config::gpio
```

6.5.1.4 mode

```
template<uint8_t number_of_channels>
uint32_t hal::Adc< number_of_channels >::Config::mode
```

6.5.1.5 prescaler

```
template<uint8_t number_of_channels>
uint32_t hal::Adc< number_of_channels >::Config::prescaler
```

6.5.1.6 rcc_clock

```
template<uint8_t number_of_channels>
rcc_periph_clken hal::Adc< number_of_channels >::Config::rcc_clock
```

6.5.1.7 rcc_reset

```
template<uint8_t number_of_channels>
rcc_periph_rst hal::Adc< number_of_channels >::Config::rcc_reset
```

6.5.1.8 resolution

```
template<uint8_t number_of_channels>
uint32_t hal::Adc< number_of_channels >::Config::resolution
```

6.5.1.9 sample_time

```
template<uint8_t number_of_channels>
uint8_t hal::Adc< number_of_channels >::Config::sample_time
```

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

- inc/hal/adc.hpp

6.6 hal::Clock::Config Struct Reference

```
#include <clock.hpp>
```

Public Attributes

- const struct rcc_clock_scale ∗ clock_scale
- uint32_t reload
- uint8_t clocksource

6.6.1 Member Data Documentation

6.6.1.1 clock_scale

```
const struct rcc_clock_scale* hal::Clock::Config::clock_scale
```

6.6.1.2 clocksource

```
uint8_t hal::Clock::Config::clocksource
```

6.6.1.3 reload

```
uint32_t hal::Clock::Config::reload
```

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

- inc/hal/clock.hpp

6.7 hal::Gpio::Config Struct Reference

```
#include <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

6.7.1 Member Data Documentation

6.7.1.1 alt_func_num

```
uint8_t hal::Gpio::Config::alt_func_num
```

6.7.1.2 mode

```
uint8_t hal::Gpio::Config::mode
```

6.7.1.3 otype

```
uint8_t hal::Gpio::Config::otype
```

6.7.1.4 pin

```
uint16_t hal::Gpio::Config::pin
```

6.7.1.5 port

```
uint32_t hal::Gpio::Config::port
```

6.7.1.6 pull_resistor

```
uint8_t hal::Gpio::Config::pull_resistor
```

6.7.1.7 rcc_clock

```
rcc_periph_clken hal::Gpio::Config::rcc_clock
```

6.7.1.8 speed

```
uint8_t hal::Gpio::Config::speed
```

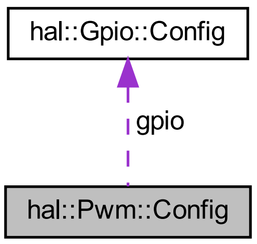
The documentation for this struct was generated from the following file:

- inc/hal/gpio.hpp

6.8 hal::Pwm::Config Struct Reference

```
#include <pwm.hpp>
```

Collaboration diagram for hal::Pwm::Config:



Public Attributes

- Gpio::Config 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

6.8.1 Member Data Documentation

6.8.1.1 clock_div

```
uint32_t hal::Pwm::Config::clock_div
```

6.8.1.2 gpio

`Gpio::Config hal::Pwm::Config::gpio`

6.8.1.3 oc_id

`tim_oc_id hal::Pwm::Config::oc_id`

6.8.1.4 oc_mode

`tim_oc_mode hal::Pwm::Config::oc_mode`

6.8.1.5 period

`uint32_t hal::Pwm::Config::period`

6.8.1.6 prescaler

`uint32_t hal::Pwm::Config::prescaler`

6.8.1.7 rcc_clock

`rcc_periph_clken hal::Pwm::Config::rcc_clock`

6.8.1.8 timer

`uint32_t hal::Pwm::Config::timer`

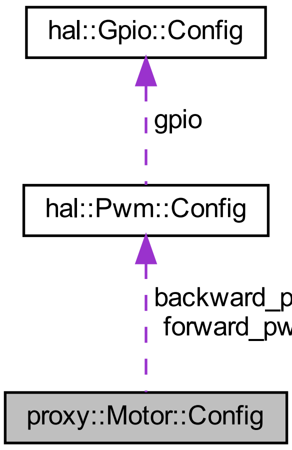
The documentation for this struct was generated from the following file:

- inc/hal/pwm.hpp

6.9 proxy::Motor::Config Struct Reference

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

Collaboration diagram for proxy::Motor::Config:



Public Attributes

- hal::Pwm::Config forward_pwm
- hal::Pwm::Config backward_pwm

6.9.1 Member Data Documentation

6.9.1.1 backward_pwm

```
hal::Pwm::Config proxy::Motor::Config::backward_pwm
```

6.9.1.2 forward_pwm

```
hal::Pwm::Config proxy::Motor::Config::forward_pwm
```

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

- inc/proxy/motor.hpp

6.10 hal::Gpio Class Reference

```
#include <gpio.hpp>
```

Classes

- struct Config

Public Member Functions

- Gpio (const Config &gpio_config)

  Construct a new 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.

6.10.1 Constructor & Destructor Documentation

6.10.1.1 Gpio()

```
hal::Gpio::Gpio (
            const Config & gpio_config )
```

Construct a new GPIO object.

Parameters

gpio_config	Configuration of the gpio instance

6.10.2 Member Function Documentation

6.10.2.1 read()

```
bool hal::Gpio::read ( ) const
```

Read the GPIO pin.

Returns

True if the pin is high, false otherwise

6.10.2.2 toggle()

```
void hal::Gpio::toggle ( )
```

Toggle the GPIO pin.

6.10.2.3 write()

```
void hal::Gpio::write (
             bool pin_state )
```

Write to the GPIO pin.

Parameters

pin_state	State of the GPIO pin

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

- inc/hal/gpio.hpp
- src/hal/gpio.cpp

6.11 proxy::LineSensors< number_of_sensors > Class Template Reference

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

Public Member Functions

- LineSensors (const typename hal::Adc< number_of_sensors >::Config &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.

6.11.1 Constructor & Destructor Documentation

6.11.1.1 LineSensors()

```
template<uint8_t number_of_sensors>
proxy::LineSensors< number_of_sensors >::LineSensors (
             const typename hal::Adc< number_of_sensors >::Config & adc_config )
```

Construct a new Line Sensors object.

Parameters

adc_config	Configuration of the ADC used to read the line sensors

6.11.2 Member Function Documentation

6.11.2.1 calibrate_black()

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

Calibrates the line sensors for the black background.

6.11.2.2 calibrate_white()

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

Calibrates the line sensors for the white line.

6.11.2.3 get_position()

```
template<uint8_t number_of_sensors>
float proxy::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

6.12 proxy::Locomotion Class Reference

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

Public Member Functions

- Locomotion (const Motor::Config &left_motor_config, const Motor::Config &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.

6.12.1 Constructor & Destructor Documentation

6.12.1.1 Locomotion()

```
proxy::Locomotion::Locomotion (
            const Motor::Config & left_motor_config,
            const Motor::Config & right_motor_config,
            float left_deadzone = 0.0,
            float right_deadzone = 0.0 )
```

Construct a new Locomotion object.

Parameters

left_motor_config	Configuration of the left motor
right_motor_config	Configuration of the right motor
left_deadzone	Deadzone of the left motor
right_deadzone	Deadzone of the right motor

6.12.2 Member Function Documentation

6.12.2.1 linear_decay()

```
float proxy::Locomotion::linear_decay (
            float angular_error,
            float dependency )  [static]
```

Compute the linear decay of the angular error.

Parameters

angular_error	Angular error
dependency	Dependency of the linear decay

Returns

Linear decay

6.12.2.2 set_speeds()

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

Set the speeds of the motors.

Parameters

linear	Linear speed
angular	Angular speed

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

- inc/proxy/locomotion.hpp
- src/proxy/locomotion.cpp

6.13 proxy::Motor Class Reference

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

Classes

- struct Config

Public Member Functions

- Motor (const Config &motor_config, float deadzone=0.0)
  Construct a new Motor object.
- void set_speed (float speed)
  Set the speed object.

Static Public Attributes

- static constexpr float max_command = 100.0

  Maximum value the motor command.
- static constexpr float min_command = -max_command

  Minimum value of the motor command.

6.13.1 Constructor & Destructor Documentation

6.13.1.1 Motor()

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

Construct a new Motor object.

Parameters

motor_config	Configuration for each pwm of the motor
deadzone	Minimum value of the pwm to start the motor

6.13.2 Member Function Documentation

6.13.2.1 set_speed()

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

Set the speed object.

Parameters

speed	Speed of the motor

6.13.3 Member Data Documentation

6.13.3.1 max_command

```
constexpr float proxy::Motor::max_command = 100.0  [static], [constexpr]
```

Maximum value the motor command.

6.13.3.2 min_command

```
constexpr float proxy::Motor::min_command = -max_command  [static], [constexpr]
```

Minimum value of the motor command.

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

- inc/proxy/motor.hpp
- src/proxy/motor.cpp

6.14 PidController Class Reference

Implementation of simple PID controller Response = Kp(error + Ki ∗ integral(error) Kd ∗ d/dt(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.

6.14.1 Detailed Description

Implementation of simple PID controller Response = Kp(error + Ki ∗ integral(error) Kd ∗ d/dt(error))

6.14.2 Constructor & Destructor Documentation

6.14.2.1 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

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

6.14.3 Member Function Documentation

6.14.3.1 reset()

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

Reset prev_error and error_acc objects.

6.14.3.2 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

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

6.14.3.3 set_setpoint()

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

Set the setpoint object.

Parameters

setpoint	Desired state

6.14.3.4 update() [1/2]

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

Update PID with new state and return response.

Parameters

state	Current value of the controlled variable

Returns

Response

6.14.3.5 update() [2/2]

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

Update PID with new state and return response.

Parameters

state	Current value of the controlled variable
state_change	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

6.15 hal::Pwm Class Reference

```
#include <pwm.hpp>
```

Classes

- struct Config

Public Member Functions

- Pwm (const Config &pwm_config)

  Construct a new Pwm object.
- void set_compare (uint32_t compare)

  Set the PWM duty cycle.

6.15.1 Constructor & Destructor Documentation

6.15.1.1 Pwm()

```
hal::Pwm::Pwm (
            const Config & pwm_config )
```

Construct a new Pwm object.

Parameters

pwm_config	Configuration for the pwm instance

6.15.2 Member Function Documentation

6.15.2.1 set_compare()

```
void hal::Pwm::set_compare (
            uint32_t compare )
```

Set the PWM duty cycle.

Parameters

compare	Value to set the duty cycle

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

- inc/hal/pwm.hpp
- src/hal/pwm.cpp

6.16 hal::Timer Class Reference

```
#include <timer.hpp>
```

Public Member Functions

- Timer ()

  Construct a new 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.

6.16.1 Constructor & Destructor Documentation

6.16.1.1 Timer()

```
hal::Timer::Timer ( )
```

Construct a new Timer object.

6.16.2 Member Function Documentation

6.16.2.1 get_time()

```
float hal::Timer::get_time (
            void ) const
```

Get elapsed time since last reset.

Returns

Elapsed time in seconds

6.16.2.2 increment_system_ticks()

```
void hal::Timer::increment_system_ticks (
            void ) [static]
```

Increment the system ticks.

6.16.2.3 reset()

```
void hal::Timer::reset (
            void )
```

Reset the timer.

6.16.2.4 sleep()

```
void hal::Timer::sleep (
            uint32_t milliseconds ) [static]
```

Sleep for a given amount of time.

Parameters

milliseconds	Time to sleep in milliseconds

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

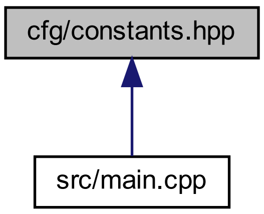
- inc/hal/timer.hpp
- src/hal/timer.cpp

Chapter 7

File Documentation

7.1 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

7.1.1 Variable Documentation

7.1.1.1 filter_frequency

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

7.1.1.2 kd

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

7.1.1.3 ki

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

7.1.1.4 kp

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

7.1.1.5 left_deadzone

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

7.1.1.6 linear_base_speed

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

7.1.1.7 linear_decay

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

7.1.1.8 max_integral

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

7.1.1.9 right_deadzone

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

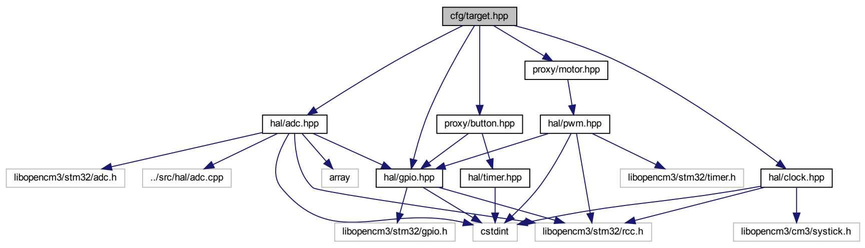
7.1.1.10 saturation

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

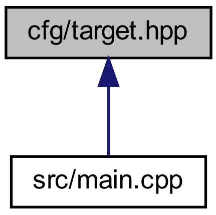
7.2 cfg/target.hpp File Reference

```
#include "hal/adc.hpp"
#include "hal/clock.hpp"
#include "hal/gpio.hpp"
#include "proxy/button.hpp"
#include "proxy/motor.hpp"
```

Include dependency graph for target.hpp:



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



Variables

- constexpr hal::Clock::Config clock_config
- constexpr hal::Gpio::Config button_config
- proxy::Button::pull_resistor_t button_pull_resistor = proxy::Button::PULL_UP
- constexpr hal::Gpio::Config led_config
- constexpr proxy::Motor::Config left_motor_config
- constexpr proxy::Motor::Config 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 hal::Adc< adc_num_channels >::Config line_sensors_config

7.2.1 Variable Documentation

7.2.1.1 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,
}
```

7.2.1.2 adc_num_channels

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

7.2.1.3 adc_readings_per_channel

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

7.2.1.4 button_config

```
constexpr hal::Gpio::Config button_config  [constexpr]
```

Initial value:

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

7.2.1.5 button_pull_resistor

```
proxy::Button::pull_resistor_t button_pull_resistor = proxy::Button::PULL_UP
```

7.2.1.6 clock_config

```
constexpr hal::Clock::Config clock_config  [constexpr]
```

Initial value:

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

7.2.1.7 led_config

```
constexpr hal::Gpio::Config 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,
}
```

7.2.1.8 left_motor_config

```
constexpr proxy::Motor::Config left_motor_config  [constexpr]
```

7.2.1.9 line_sensors_config

```
constexpr hal::Adc<adc_num_channels>::Config 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,
}
```

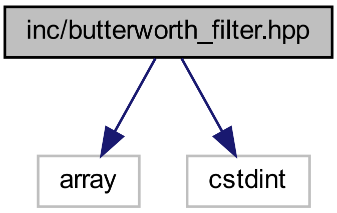
7.2.1.10 right_motor_config

```
constexpr proxy::Motor::Config right_motor_config  [constexpr]
```

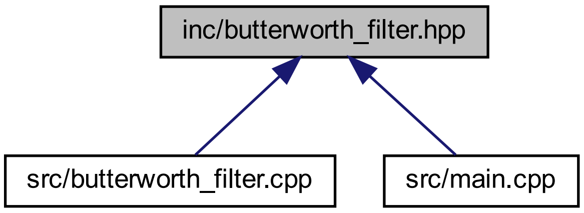
7.3 inc/butterworth_filter.hpp File Reference

```
#include <array>
#include <cstdint>
```

Include dependency graph for butterworth_filter.hpp:



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



Classes

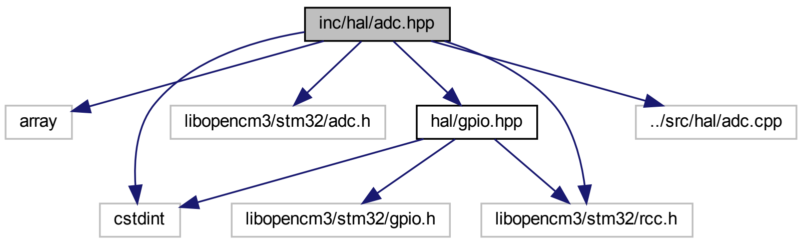
- class ButterworthFilter

  Implementation of Butterworth second order low-pass filter A generic digital filter follows the relation a0 ∗ y[k] = sum(bi ∗ x[k - i]) - sum(aj ∗ 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 continuos coefficients of the Butterworth filter is available here: `https://en.wikipedia.org/wiki/Butterworth_filter` *The discrete version were computed with the Tustin method:* `https://en.wikipedia.org/wiki/Bilinear_transform`.

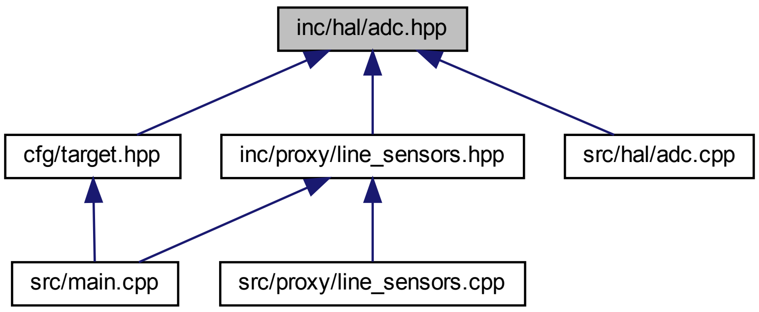
7.4 inc/hal/adc.hpp File Reference

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

Include dependency graph for adc.hpp:



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



Classes

- class hal::Adc< number_of_channels >
- struct hal::Adc< number_of_channels >::Config

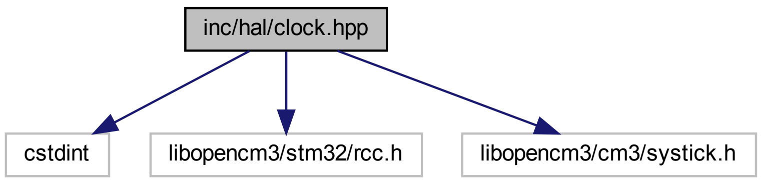
Namespaces

- hal

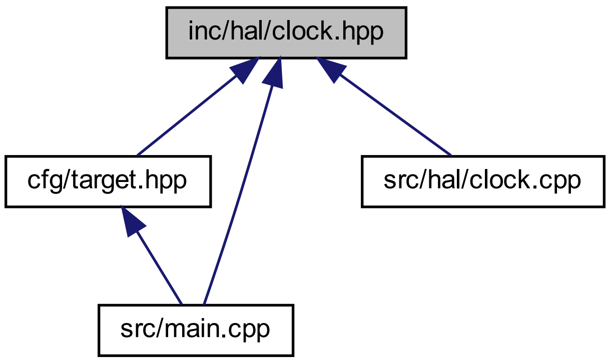
7.5 inc/hal/clock.hpp File Reference

```
#include <cstdint>
#include <libopencm3/stm32/rcc.h>
#include <libopencm3/cm3/systick.h>
```

Include dependency graph for clock.hpp:



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



Classes

- class hal::Clock
- struct hal::Clock::Config

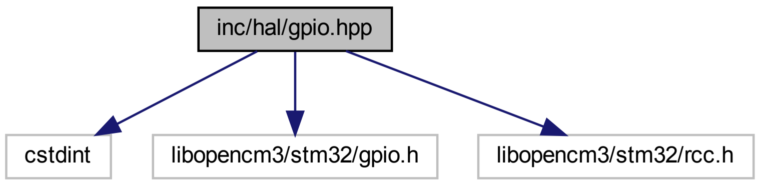
Namespaces

- hal

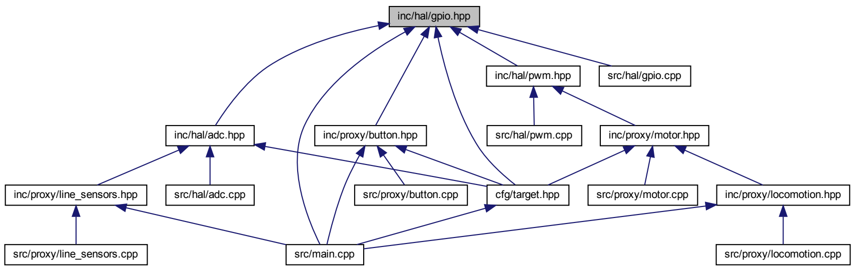
7.6 inc/hal/gpio.hpp File Reference

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

Include dependency graph for gpio.hpp:



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



Classes

- class hal::Gpio
- struct hal::Gpio::Config

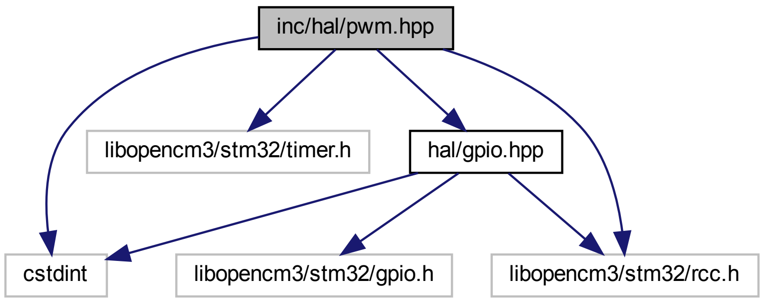
Namespaces

- hal

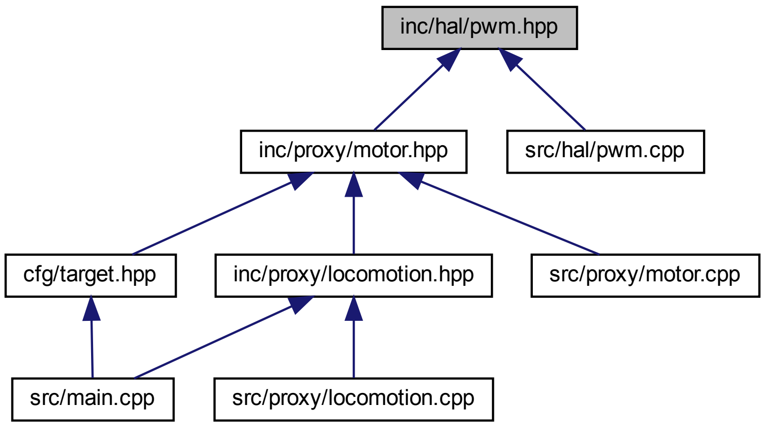
7.7 inc/hal/pwm.hpp File Reference

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

Include dependency graph for pwm.hpp:



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



Classes

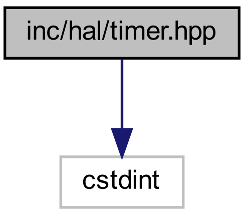
- class hal::Pwm
- struct hal::Pwm::Config

Namespaces

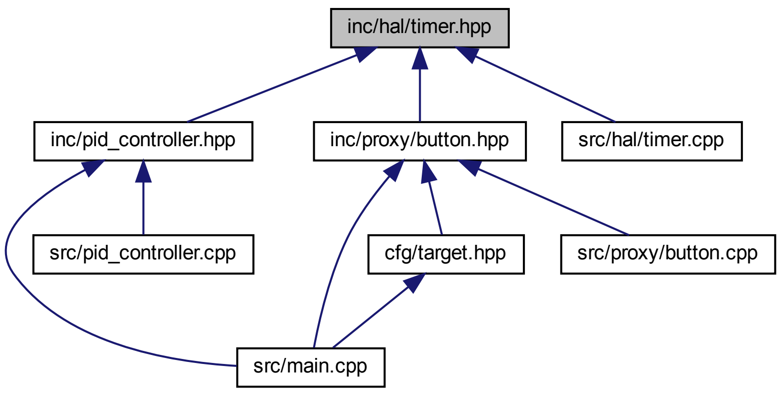
- hal

7.8 inc/hal/timer.hpp File Reference

```
#include <cstdint>
```
Include dependency graph for timer.hpp:



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



Classes

- class hal::Timer

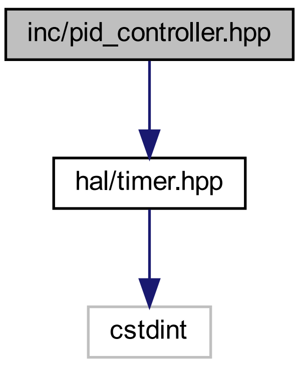
Namespaces

- hal

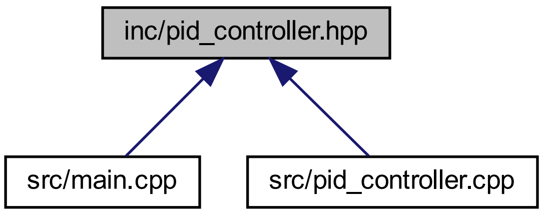
7.9 inc/pid_controller.hpp File Reference

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

Include dependency graph for pid_controller.hpp:



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



Classes

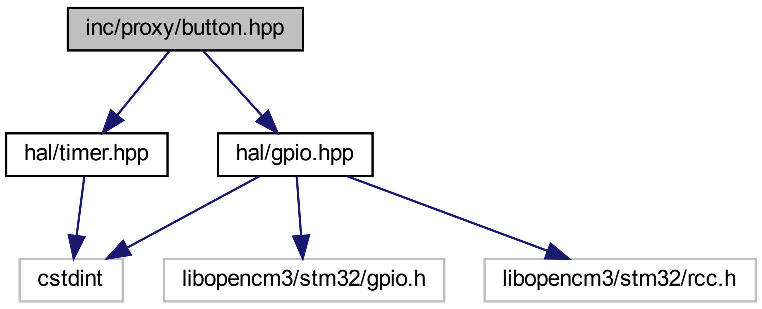
- class PidController

  Implementation of simple PID controller Response = Kp(error + Ki ∗ integral(error) Kd ∗ d/dt(error))

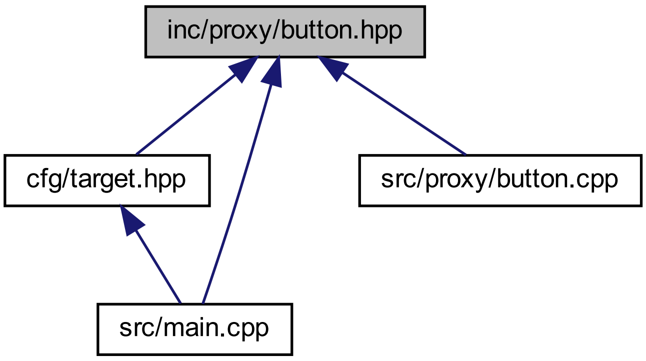
7.10 inc/proxy/button.hpp File Reference

```
#include "hal/gpio.hpp"
#include "hal/timer.hpp"
```

Include dependency graph for button.hpp:



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



Classes

- class proxy::Button

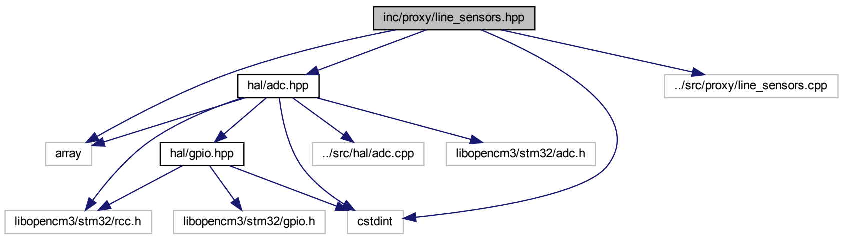
Namespaces

- proxy

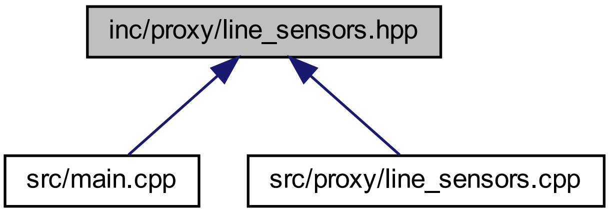
7.11 inc/proxy/line_sensors.hpp File Reference

```
#include <array>
#include <cstdint>
#include "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 proxy::LineSensors< number_of_sensors >

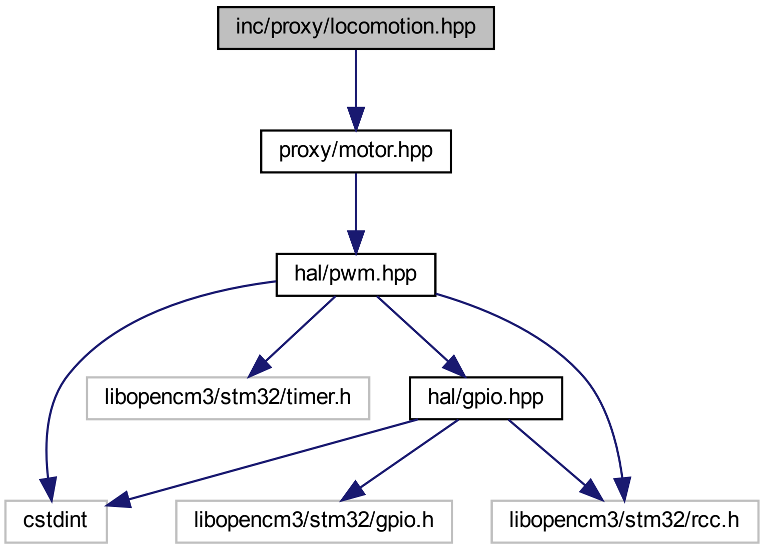
Namespaces

- proxy

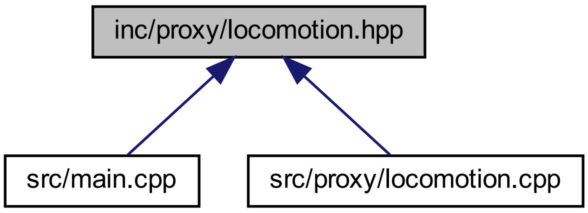
7.12 inc/proxy/locomotion.hpp File Reference

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

Include dependency graph for locomotion.hpp:



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



Classes

- class proxy::Locomotion

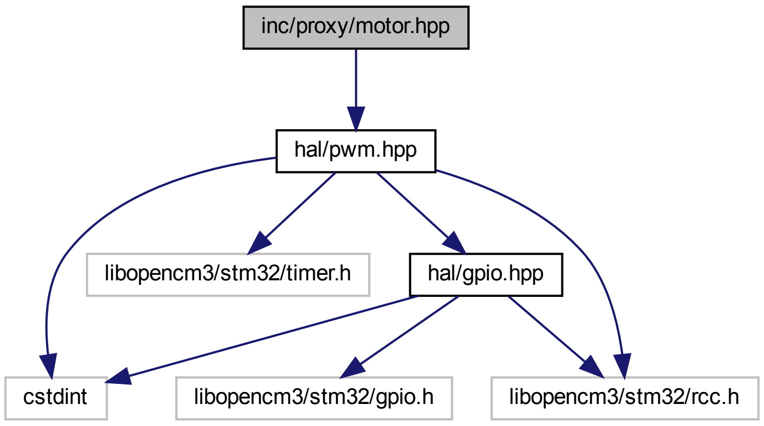
Namespaces

- proxy

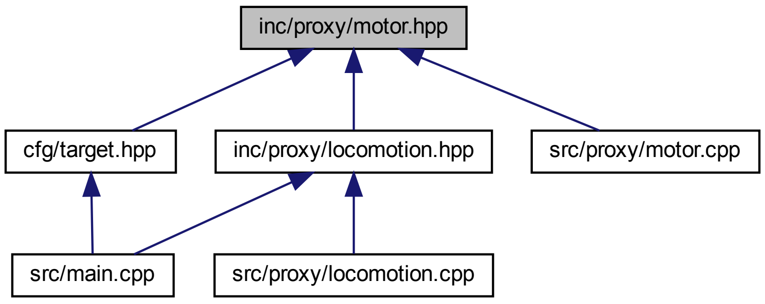
7.13 inc/proxy/motor.hpp File Reference

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

Include dependency graph for motor.hpp:



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



Classes

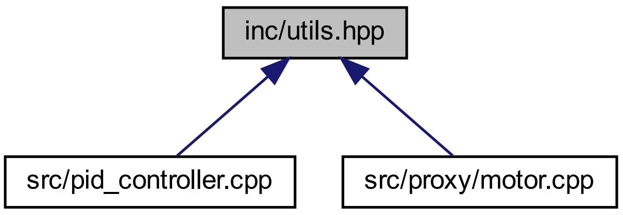
- class proxy::Motor
- struct proxy::Motor::Config

Namespaces

- proxy

7.14 inc/utils.hpp File Reference

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



Functions

- constexpr float constrain (float value, float min, float max)
- constexpr float map (float value, float from_min, float from_max, float to_min, float to_max)

7.14.1 Function Documentation

7.14.1.1 constrain()

```
constexpr float constrain (
            float value,
            float min,
            float max )  [constexpr]
```

7.14.1.2 map()

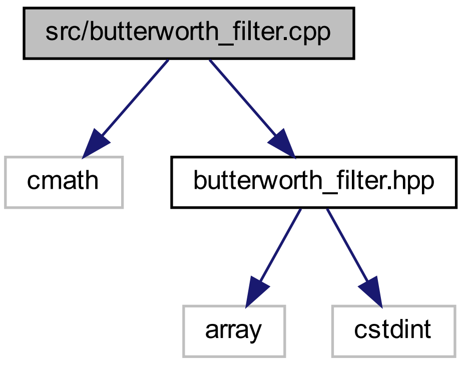
```
constexpr float map (
            float value,
            float from_min,
            float from_max,
            float to_min,
            float to_max )  [constexpr]
```

7.15 README.md File Reference

7.16 src/butterworth_filter.cpp File Reference

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

Include dependency graph for butterworth_filter.cpp:



Variables

- constexpr float srqt2 = 1.41421356237309504

7.16.1 Variable Documentation

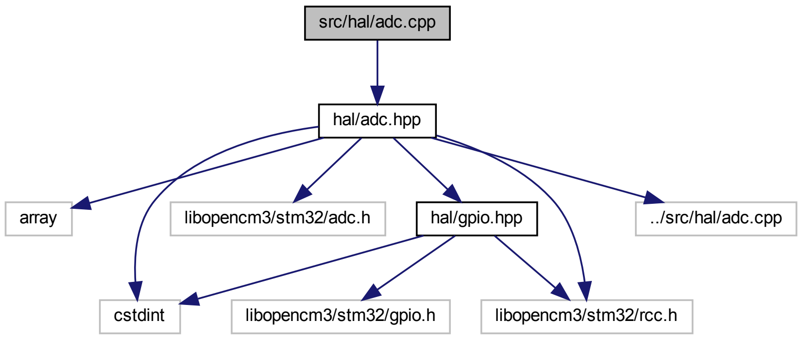
7.16.1.1 srqt2

```
constexpr float srqt2 = 1.41421356237309504  [constexpr]
```

7.17 src/hal/adc.cpp File Reference

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

Include dependency graph for adc.cpp:



Namespaces

- hal

Macros

- #define __ADC_CPP__

7.17.1 Macro Definition Documentation

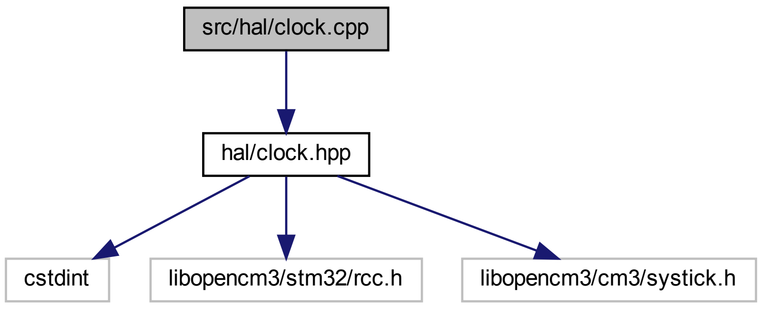
7.17.1.1 __ADC_CPP__

```
#define __ADC_CPP__
```

7.18 src/hal/clock.cpp File Reference

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

Include dependency graph for clock.cpp:



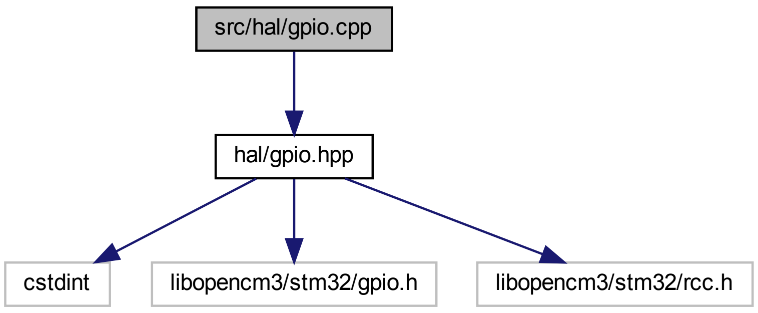
Namespaces

- hal

7.19 src/hal/gpio.cpp File Reference

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

Include dependency graph for gpio.cpp:



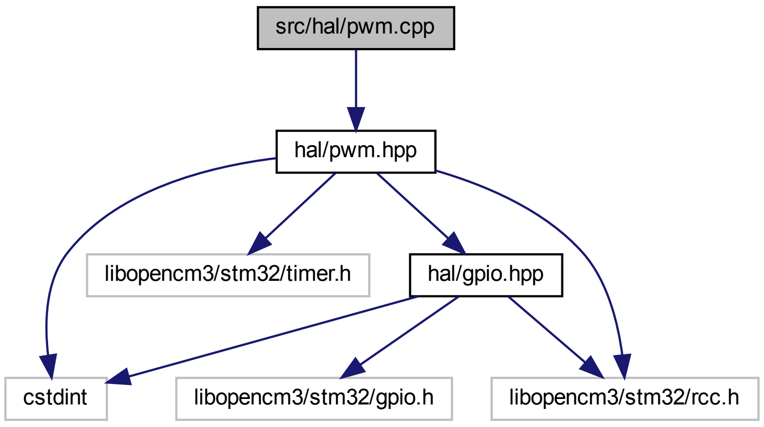
Namespaces

- hal

7.20 src/hal/pwm.cpp File Reference

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

Include dependency graph for pwm.cpp:



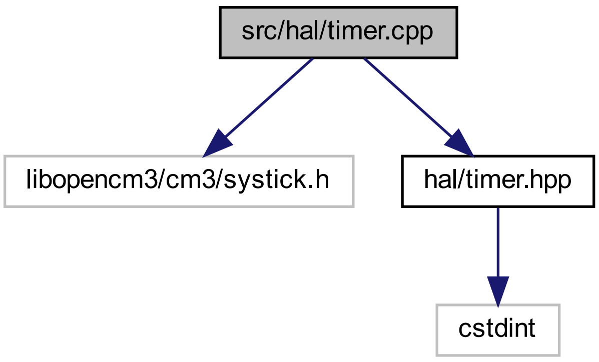
Namespaces

- hal

7.21 src/hal/timer.cpp File Reference

```
#include <libopencm3/cm3/systick.h>
#include "hal/timer.hpp"
```

Include dependency graph for timer.cpp:



Namespaces

- hal

Functions

- void sys_tick_handler (void)

7.21.1 Function Documentation

7.21.1.1 sys_tick_handler()

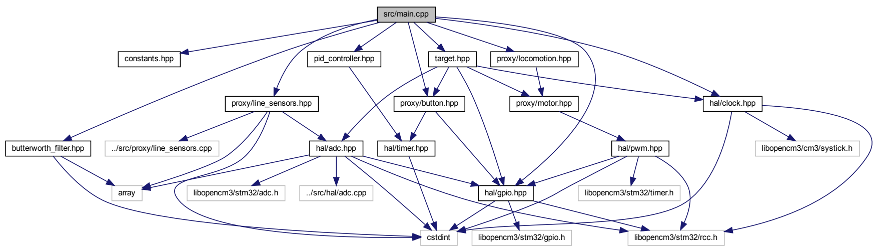
```
void sys_tick_handler (
            void  )
```

7.22 src/main.cpp File Reference

```
#include "constants.hpp"
#include "target.hpp"
#include "butterworth_filter.hpp"
#include "pid_controller.hpp"
#include "hal/clock.hpp"
#include "hal/gpio.hpp"
#include "proxy/button.hpp"
#include "proxy/line_sensors.hpp"
```

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

Include dependency graph for main.cpp:



Functions

- int main (void)

7.22.1 Function Documentation

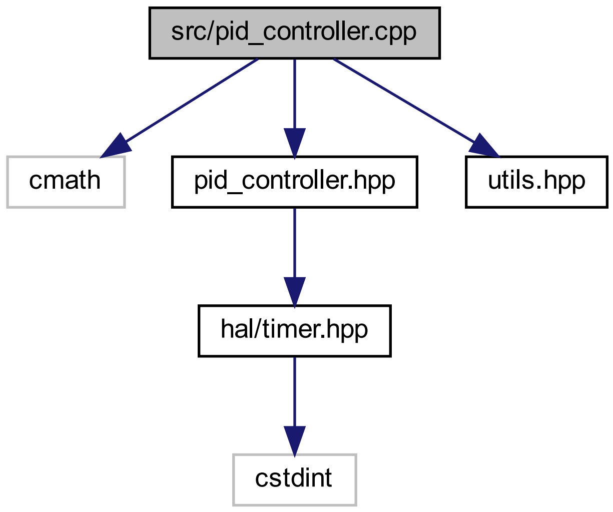
7.22.1.1 main()

```
int main (
            void )
```

7.23 src/pid_controller.cpp File Reference

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

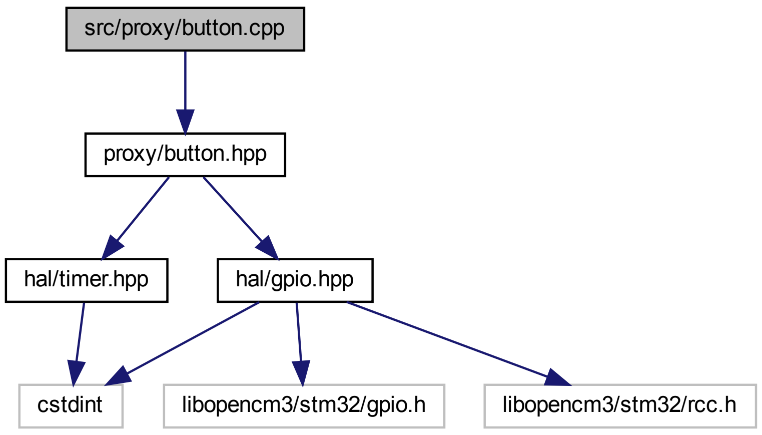
Include dependency graph for pid_controller.cpp:



7.24 src/proxy/button.cpp File Reference

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

Include dependency graph for button.cpp:



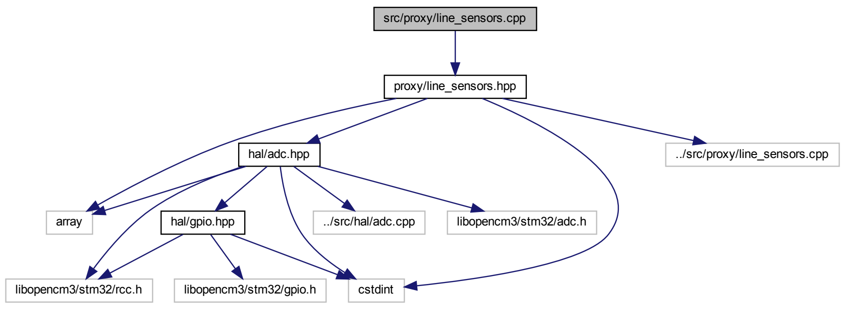
Namespaces

- proxy

7.25 src/proxy/line_sensors.cpp File Reference

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

Include dependency graph for line_sensors.cpp:



Namespaces

- proxy

Macros

- #define __LINE_SENSORS_CPP__

Variables

- constexpr uint32_t proxy::default_white_value = 3850
- constexpr uint32_t proxy::default_black_value = 4000

7.25.1 Macro Definition Documentation

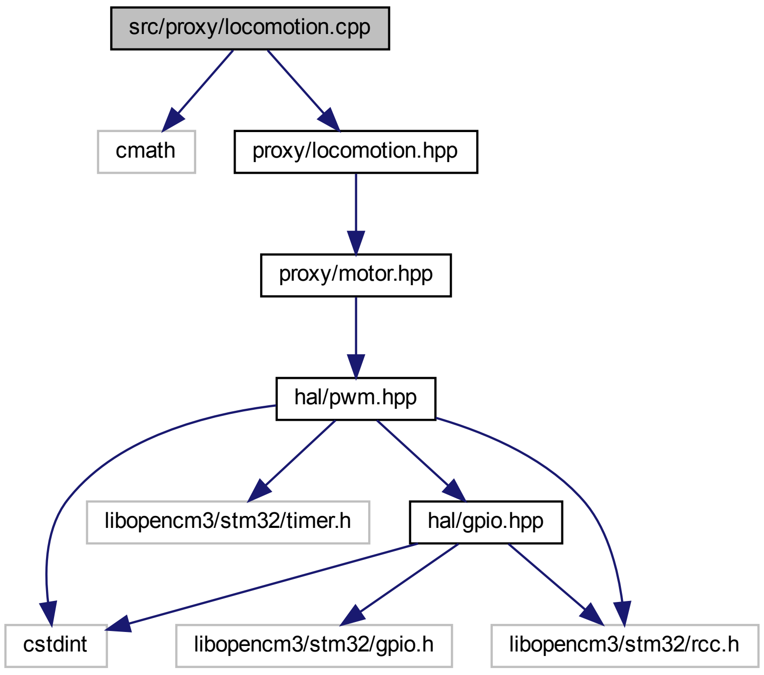
7.25.1.1 __LINE_SENSORS_CPP__

```
#define __LINE_SENSORS_CPP__
```

7.26 src/proxy/locomotion.cpp File Reference

```
#include <cmath>
#include "proxy/locomotion.hpp"
```

Include dependency graph for locomotion.cpp:



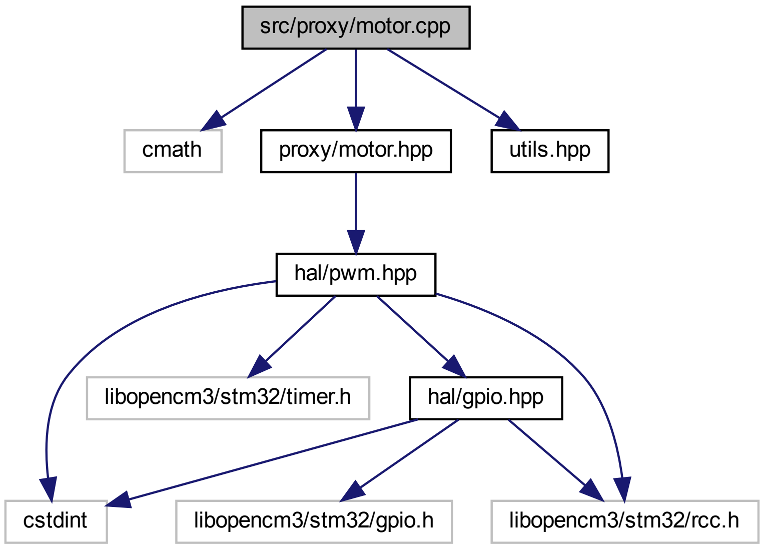
Namespaces

- proxy

7.27 src/proxy/motor.cpp File Reference

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

Include dependency graph for motor.cpp:



Namespaces

- proxy

Index