Jimmy Challenge

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3   Data Structure Documentation

3.1   interface::Audio Class Reference

```
#include <Audio.h>
```

Inheritance diagram for interface::Audio:
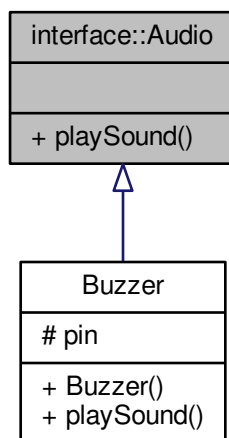
Collaboration diagram for interface::Audio:
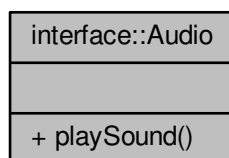
Public Member Functions

- virtual void **playSound** (int)=0

3.1.1 Detailed Description

Interface for audio devices

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

- src/interface/Audio.h

3.2 Button Class Reference

```
#include <Button.h>
```

Inheritance diagram for Button:

```
┌─────────────────────────┐
│     interface::Input    │
├─────────────────────────┤
│                         │
├─────────────────────────┤
│  + readBool()           │
│  + readDistance()       │
└─────────────────────────┘
             △
             │
┌─────────────────────────┐
│         Button          │
├─────────────────────────┤
│  # pin                  │
├─────────────────────────┤
│  + Button()             │
│  + readBool()           │
└─────────────────────────┘
```

Collaboration diagram for Button:

```
┌─────────────────────────┐
│     interface::Input    │
├─────────────────────────┤
│                         │
├─────────────────────────┤
│  + readBool()           │
│  + readDistance()       │
└─────────────────────────┘
             △
             │
┌─────────────────────────┐
│         Button          │
├─────────────────────────┤
│  # pin                  │
├─────────────────────────┤
│  + Button()             │
│  + readBool()           │
└─────────────────────────┘
```
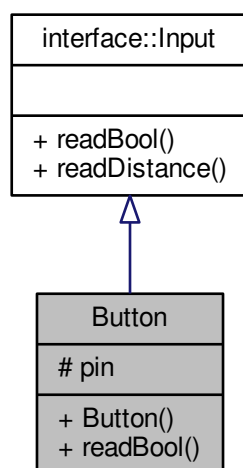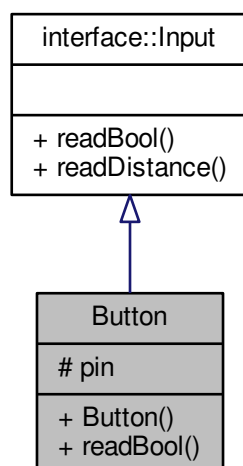
Public Member Functions

- Button (int, unsigned long)

Initialize the button and set init config

- bool readBool ()

    Read a state from the hardware button

Protected Attributes

- int **pin**

3.2.1   Detailed Description

Class to manage signals from a button

3.2.2   Member Function Documentation

3.2.2.1   bool Button::readBool ( ) `[virtual]`

Read a state from the hardware button

When the button is pressed read the state and avoid debouncing

Reimplemented from interface::Input.

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

- src/input/Button.h
- src/input/Button.cpp

3.3   ButtonTask Class Reference

```
#include <ButtonTask.h>
```

Inheritance diagram for ButtonTask:

```
┌─────────────────────────────────┐
│              Task               │
├─────────────────────────────────┤
│                                 │
├─────────────────────────────────┤
│ + init()                        │
│ + tick()                        │
│ + isEnabled()                   │
│ + enable()                      │
│ + disable()                     │
│ + updateAndCheckTime()          │
│ # lambdaTick()                  │
└─────────────────────────────────┘
                 △
                 │
┌─────────────────────────────────┐
│           ButtonTask            │
├─────────────────────────────────┤
│ + btn                           │
│ # pin                           │
│ # debounceDelay                 │
├─────────────────────────────────┤
│ + ButtonTask()                  │
│ + init()                        │
│ + tick()                        │
└─────────────────────────────────┘
```

Collaboration diagram for ButtonTask:
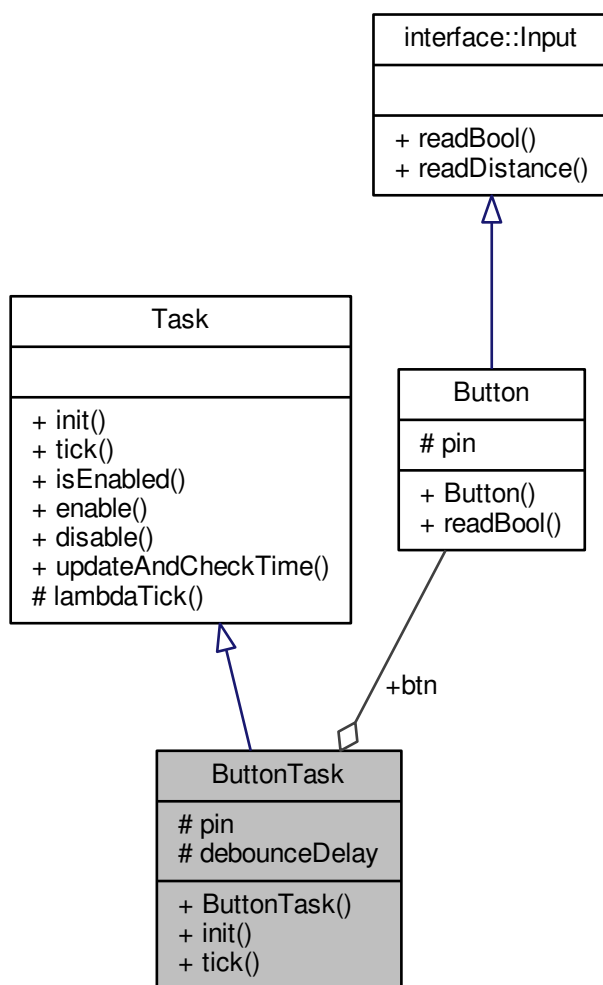
Public Member Functions

- ButtonTask (int, unsigned long, Context ∗)

    Init pin, debounce time and context
- void init (int, void(∗)())

    Set the period time for the task execution and the behaviour
- void tick ()

    The function to execute when the scheduler gives the resources to the task

Data Fields

- Button ∗ **btn**

Protected Attributes

- int **pin**
- unsigned long **debounceDelay**

Additional Inherited Members

3.3.1 Detailed Description

Class to create a task using the state of a button

The documentation for this class was generated from the following files:
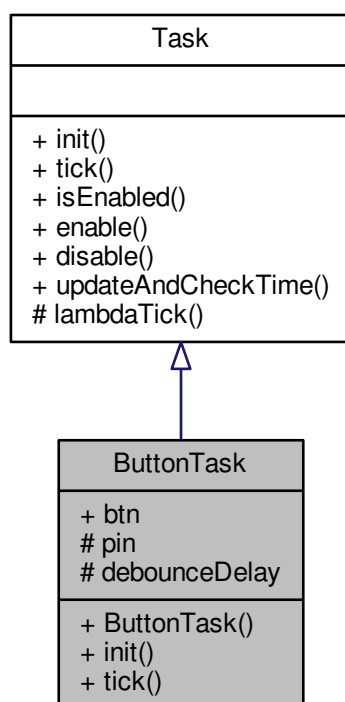
- src/task/ButtonTask.h
- src/task/ButtonTask.cpp

3.4 Buzzer Class Reference

```
#include <Buzzer.h>
```

Inheritance diagram for Buzzer:

Collaboration diagram for Buzzer:

```
┌─────────────────────┐
│  interface::Audio   │
├─────────────────────┤
│                     │
├─────────────────────┤
│  + playSound()      │
└─────────────────────┘
          △
          │
┌─────────────────────┐
│      Buzzer         │
├─────────────────────┤
│  # pin              │
├─────────────────────┤
│  + Buzzer()         │
│  + playSound()      │
└─────────────────────┘
```

Public Member Functions

- Buzzer (const int)

    Set the buzzer
- void playSound (int)

    Play a sound

Protected Attributes

- int **pin**

3.4.1   Detailed Description

Class to manage the buzzer sounds

3.4.2   Member Function Documentation

3.4.2.1   void Buzzer::playSound (int *sound*) `[virtual]`

Play a sound

Choose wich sound to play

Parameters

in	*sound*	The sound to play

Implements interface::Audio.

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

- src/output/Buzzer.h
- src/output/Buzzer.cpp

## 3.5 BuzzerTask Class Reference

```
#include <BuzzerTask.h>
```

Inheritance diagram for BuzzerTask:
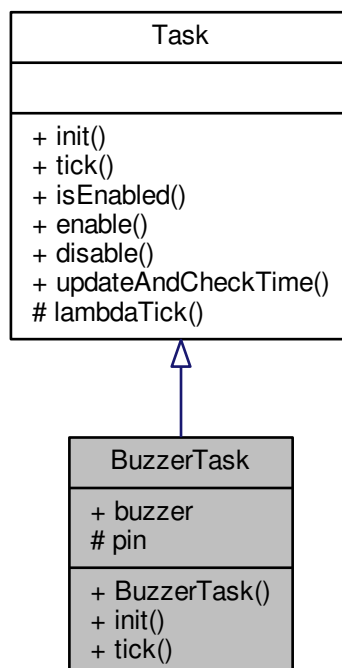
Collaboration diagram for BuzzerTask:
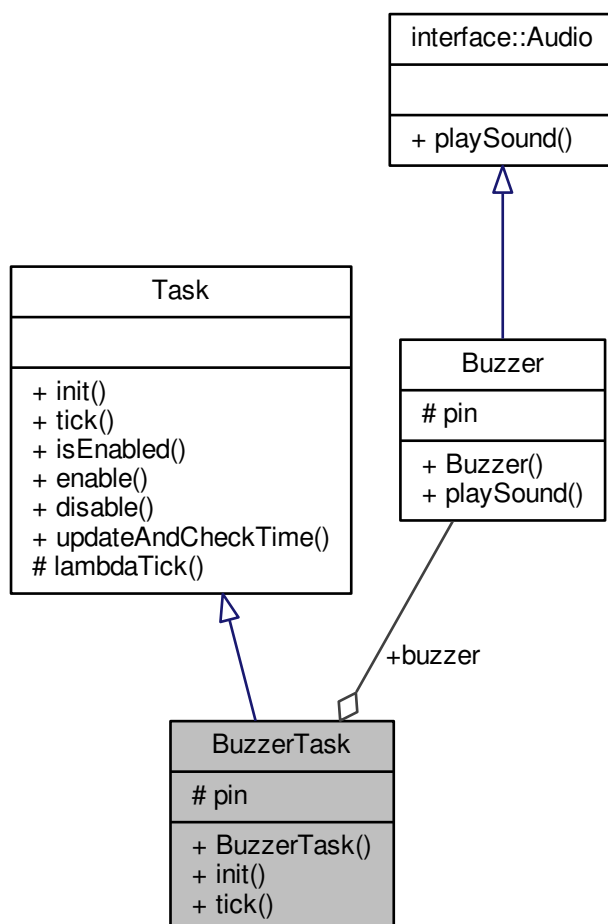
Public Member Functions

- BuzzerTask (int, Context ∗)

  Init pin and context
- void init (int, void(∗)())

  Set the period time for the task execution and the behaviour
- void tick ()

  The function to execute when the scheduler gives the resources to the task

Data Fields

- Buzzer ∗ **buzzer**

Protected Attributes

- int **pin**

Additional Inherited Members

3.5.1   Detailed Description
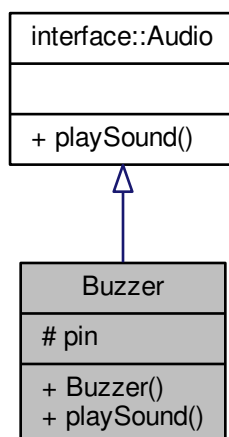
Class to manage the behaviour of a buzzer

The documentation for this class was generated from the following files:
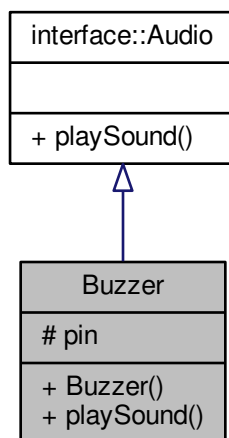
- src/task/BuzzerTask.h
- src/task/BuzzerTask.cpp

3.6   Context Class Reference

```
#include <Context.h>
```

Collaboration diagram for Context:
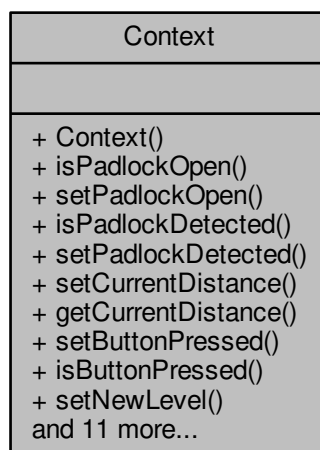
Public Member Functions

- Context (int maxDistance, Multiplexer ∗mux)

    Construct and initialize the 'Context'
- bool isPadlockOpen ()

    If the player has guessed and opened the padlock
- void setPadlockOpen (bool padlockOpen)

    Set the padlock's state
- bool isPadlockDetected ()

    If the player has guessed the secret position
- void setPadlockDetected (bool padlockDetected)

> Set the state of the padlock when user guess the secret position

- void setCurrentDistance (int currentDistance)

  > Set the distance at which the padlock will open

- int getCurrentDistance ()

  > Get the distance at which the padlock will open

- void setButtonPressed (bool buttonPressed)

  > Set the state of the button if pressed

- bool isButtonPressed ()

  > Get the state of the button

- void setNewLevel ()

  > Set a new level

- uint8_t getDelta ()

  > Get the margin of error for the distance

- uint8_t getLevel ()

  > Get the level to play

- int getSecret ()

  > Get the scret distance where the padlock will open

- void newRandomNumber ()

  > Generate a new random number

- void setGameOver (bool gameOver)

  > Set the game state

- bool isGameOver ()

  > Get the stat of the game

- void setDangerLevel (uint8_t dangerLevel)

  > Set the level at which the padlock will starts to break

- uint8_t getDangerLevel ()

  > Get the current level of breakage of padlock

- void setLockpicking (bool state)

  > Set if the padlock is found and the user starts to pick

- bool isLockpicking ()

  > Get if the user start to lock-picking the padlock

- void carousel (uint8_t delay1, uint8_t delay2)

  > Run a carousel with two led color

3.6.1 Detailed Description

Contains all status variable for the program.

It's used to share informations between task and coordinate them creating a simple game.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 Context::Context (int *maxDistance,* **Multiplexer** ∗ *mux*) `[inline]`

Construct and initialize the '**Context**'

Initialize all parameters of the game and set the random seed with analog output entropy

Parameters

in	*maxDistance*	Maximum distance game
in	*mux*	Multiplexer instance

3.6.3 Member Function Documentation

3.6.3.1 void Context::carousel (uint8_t *delay1,* **uint8_t** *delay2* **)** `[inline]`

Run a carousel with two led color

The carousel activate two circular array of 6 leds.

Parameters

in	*delay1*	The delay before change position
in	*delay2*	The delay before change position

3.6.3.2 void Context::setButtonPressed (bool *buttonPressed* **)** `[inline]`

Set the state of the button if pressed

Parameters

in	*buttonPressed*	The state of the button

3.6.3.3 void Context::setCurrentDistance (int *currentDistance* **)** `[inline]`

Set the distance at which the padlock will open

Parameters

in	*currentDistance*	The distance to set

3.6.3.4 void Context::setGameOver (bool *gameOver* **)** `[inline]`

Set the game state

Parameters

in	*gameOver*	The state of the game

3.6.3.5 void Context::setPadlockDetected (bool *padlockDetected* **)** `[inline]`

Set the state of the padlock when user guess the secret position

Parameters

in	*padlockDetected*	The state of the guessing part

3.6.3.6   void Context::setPadlockOpen (bool *padlockOpen*) `[inline]`

Set the padlock's state

Parameters

in	*padlockOpen*	The state of the padlock

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

- src/control/Context.h

3.7   interface::Input Class Reference

`#include <Input.h>`

Inheritance diagram for interface::Input:

Collaboration diagram for interface::Input:

```
┌─────────────────────────┐
│     interface::Input     │
├─────────────────────────┤
│                         │
├─────────────────────────┤
│ + readBool()            │
│ + readDistance()        │
└─────────────────────────┘
```
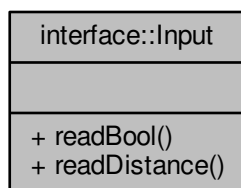
Public Member Functions

- virtual bool **readBool** ()
- virtual int **readDistance** ()

3.7.1 Detailed Description

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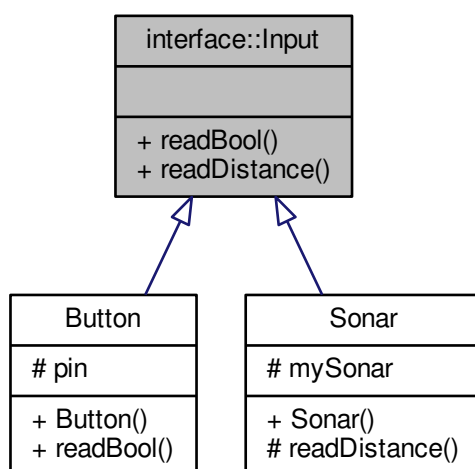
The documentation for this class was generated from the following file:

- src/interface/Input.h

3.8 Led Class Reference

```
#include <Led.h>
```

Inheritance diagram for Led:

Collaboration diagram for Led:

```
              ┌─────────────────────┐
              │   interface::Light  │
              ├─────────────────────┤
              │                     │
              ├─────────────────────┤
              │  + switchOn()       │
              │  + switchOff()      │
              └─────────────────────┘
                        △
                        │
              ┌─────────────────────┐
              │         Led         │
              ├─────────────────────┤
              │  # pin              │
              ├─────────────────────┤
              │  + Led()            │
              │  # switchOn()       │
              │  # switchOff()      │
              └─────────────────────┘
```
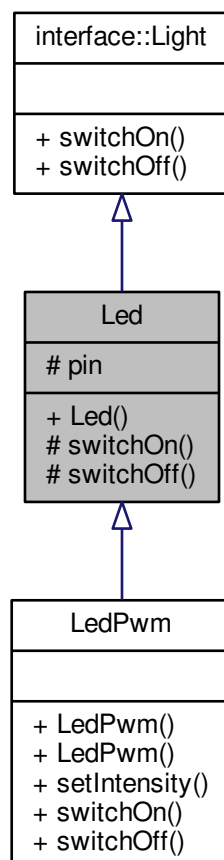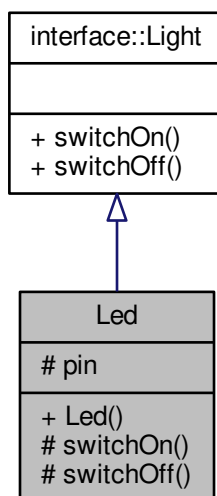
Public Member Functions

- Led (int)

    Config the led's pin

Protected Member Functions

- void switchOn ()

    Turn the led on
- void switchOff ()

    Turn the led off

Protected Attributes

- int **pin**

3.8.1 Detailed Description

Class to manage a led

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

- src/output/Led.h
- src/output/Led.cpp
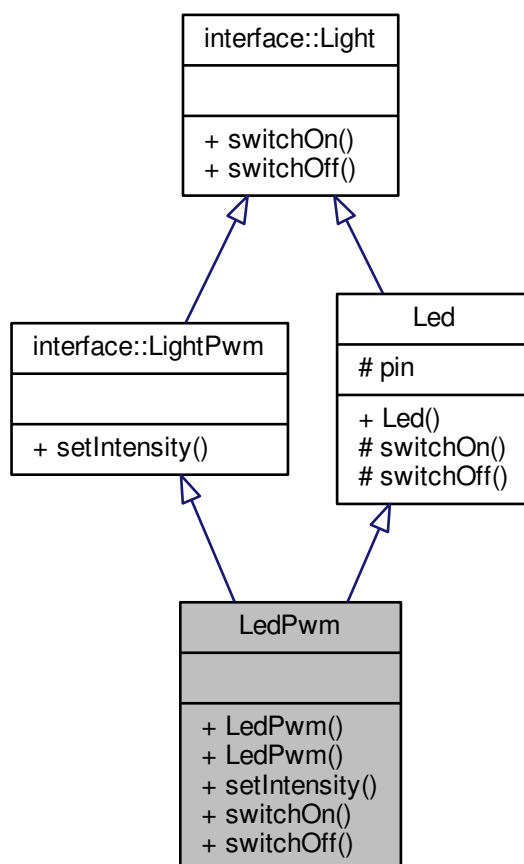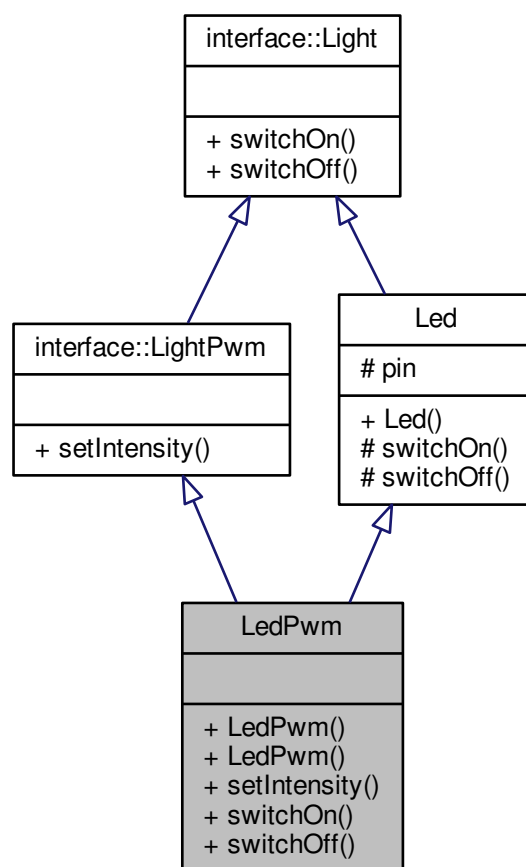
3.9 LedPwm Class Reference

```
#include <LedPwm.h>
```

Inheritance diagram for LedPwm:

Collaboration diagram for LedPwm:

```
                        ┌─────────────────────┐
                        │   interface::Light  │
                        ├─────────────────────┤
                        │                     │
                        ├─────────────────────┤
                        │  + switchOn()       │
                        │  + switchOff()      │
                        └─────────────────────┘
                          △                △
                         ╱                  ╲
            ┌──────────────────────┐    ┌──────────────────┐
            │ interface::LightPwm   │    │       Led        │
            ├──────────────────────┤    ├──────────────────┤
            │                      │    │  # pin           │
            ├──────────────────────┤    ├──────────────────┤
            │  + setIntensity()    │    │  + Led()         │
            └──────────────────────┘    │  # switchOn()    │
                       △                │  # switchOff()   │
                        ╲               └──────────────────┘
                         ╲                  △
                          ╲                ╱
                        ┌─────────────────────┐
                        │       LedPwm        │
                        ├─────────────────────┤
                        │                     │
                        ├─────────────────────┤
                        │  + LedPwm()         │
                        │  + LedPwm()         │
                        │  + setIntensity()   │
                        │  + switchOn()       │
                        │  + switchOff()      │
                        └─────────────────────┘
```

Public Member Functions

- LedPwm (int)

  Config the led's pin
- LedPwm (int, int)

  Config the led's pin and init the intensity of the led
- void setIntensity (uint8_t)

  Set the intensity of the pwm led
- void switchOn ()

  Turn the led on
- void switchOff ()

  Turn the led off

Additional Inherited Members

3.9.1 Detailed Description
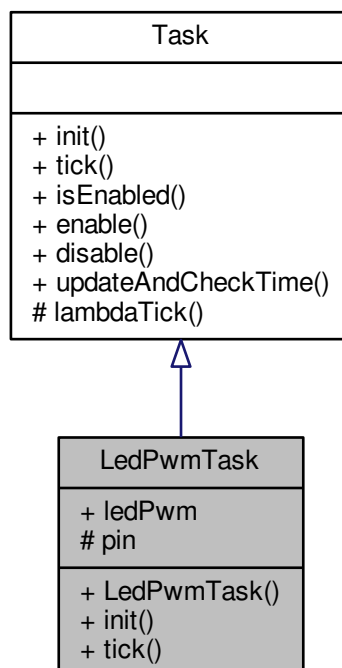
Class to manage a led with support for pwm

The documentation for this class was generated from the following files:
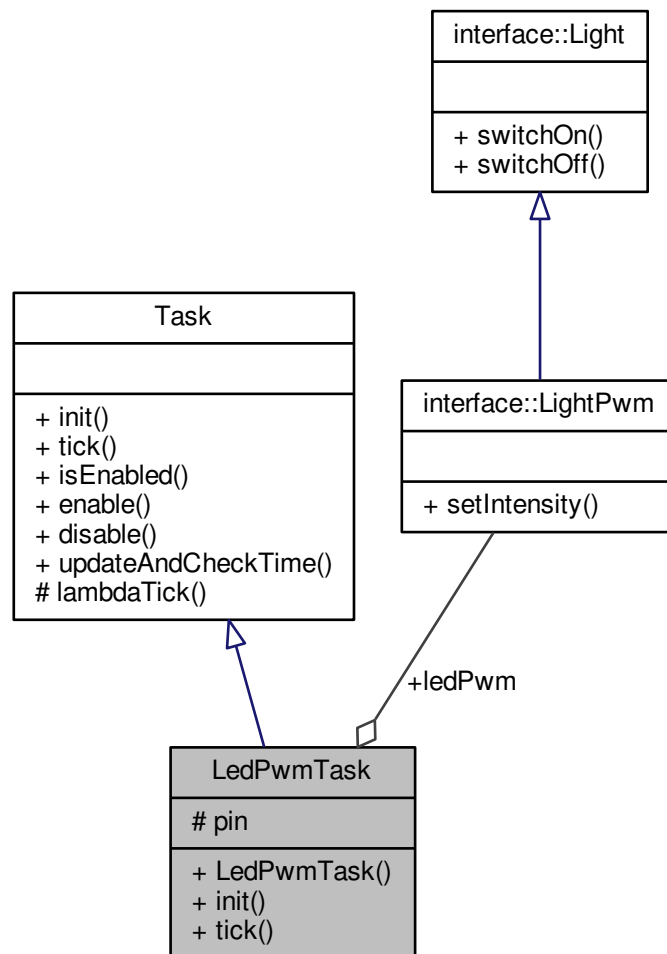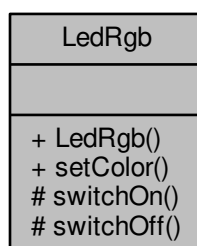
- src/output/LedPwm.h
- src/output/LedPwm.cpp

3.10 LedPwmTask Class Reference

```
#include <LedPwmTask.h>
```

Inheritance diagram for LedPwmTask:

```
┌─────────────────────────────────┐
│              Task               │
├─────────────────────────────────┤
│                                 │
├─────────────────────────────────┤
│ + init()                        │
│ + tick()                        │
│ + isEnabled()                   │
│ + enable()                      │
│ + disable()                     │
│ + updateAndCheckTime()          │
│ # lambdaTick()                  │
└─────────────────────────────────┘
                 △
                 │
┌─────────────────────────────────┐
│           LedPwmTask            │
├─────────────────────────────────┤
│ + ledPwm                        │
│ # pin                           │
├─────────────────────────────────┤
│ + LedPwmTask()                  │
│ + init()                        │
│ + tick()                        │
└─────────────────────────────────┘
```

Collaboration diagram for LedPwmTask:

Public Member Functions

- LedPwmTask (int, Context ∗)

    Init pin and context

- void init (int, void(∗)())

    Set the period time for the task execution and the behaviour

- void tick ()

    The function to execute when the scheduler gives the resources to the task

Data Fields

- interface::LightPwm ∗ **ledPwm**

Protected Attributes

- int **pin**

Additional Inherited Members

3.10.1 Detailed Description

Class to manage the behaviour of a led pwm

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

- src/task/LedPwmTask.h
- src/task/LedPwmTask.cpp

3.11 LedRgb Class Reference

```
#include <LedRgb.h>
```

Collaboration diagram for LedRgb:

```
┌─────────────────────┐
│        LedRgb        │
├─────────────────────┤
│                     │
├─────────────────────┤
│ + LedRgb()          │
│ + setColor()        │
│ # switchOn()        │
│ # switchOff()       │
└─────────────────────┘
```

Public Member Functions

- LedRgb (int, int, int)

    Config the led's pin
- void setColor (int, int, int)

    Set the color of the RGB led with a (red, green, blue) value

Protected Member Functions

- void switchOn ()

    Turn the led on
- void switchOff ()

> **Turn the led off**

3.11.1   Detailed Description
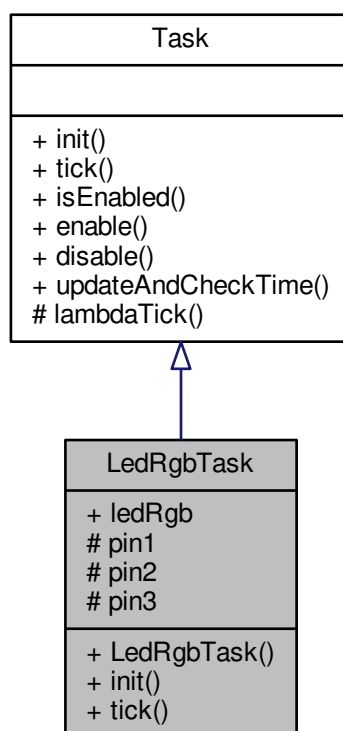
Class to manage a RGB led

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

- src/output/LedRgb.h
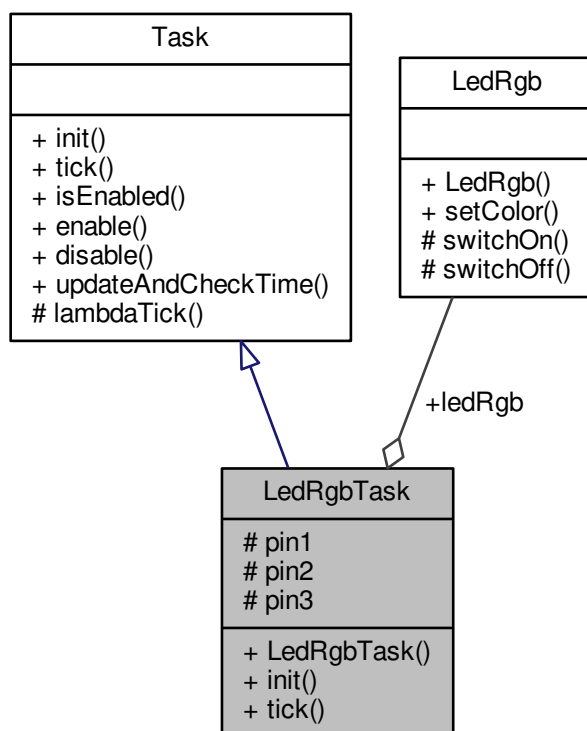- src/output/LedRgb.cpp

3.12   LedRgbTask Class Reference

```
#include <LedRgbTask.h>
```

Inheritance diagram for LedRgbTask:

Collaboration diagram for LedRgbTask:

Public Member Functions

- LedRgbTask (int, int, int, Context ∗)

  Init pins and context
- void init (int, void(∗)())

  Set the period time for the task execution and the behaviour
- void tick ()

  The function to execute when the scheduler gives the resources to the task

Data Fields

- LedRgb ∗ **ledRgb**

Protected Attributes

- int **pin1**
- int **pin2**
- int **pin3**

Additional Inherited Members

3.12.1   Detailed Description

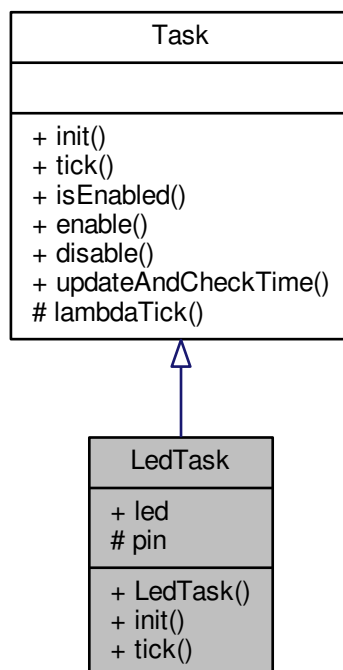Class to manage the behaviour of a RGB led

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

- src/task/LedRgbTask.h
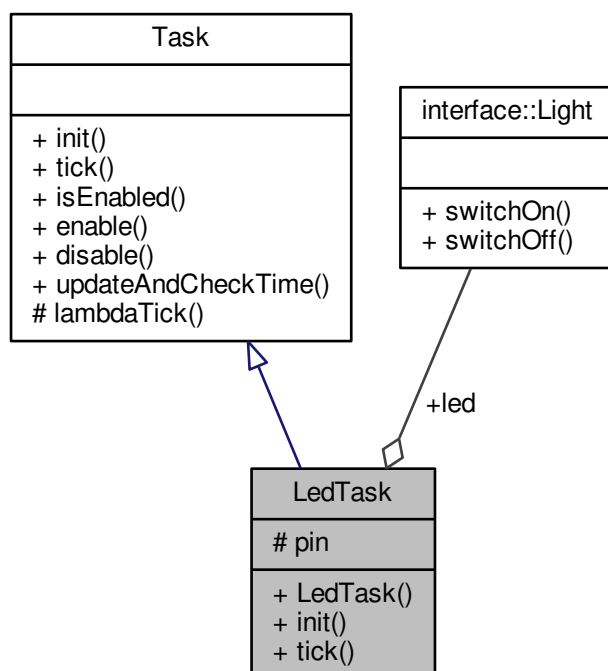- src/task/LedRgbTask.cpp

3.13   LedTask Class Reference

```
#include <LedTask.h>
```

Inheritance diagram for LedTask:

Collaboration diagram for LedTask:

Public Member Functions

- LedTask (int, Context ∗)

    Init pin and context
- void init (int, void(∗)())

    Set the period time for the task execution and the behaviour
- void tick ()

    The function to execute when the scheduler gives the resources to the task

Data Fields

- interface::Light ∗ **led**

Protected Attributes

- int **pin**

Additional Inherited Members

3.13.1 Detailed Description
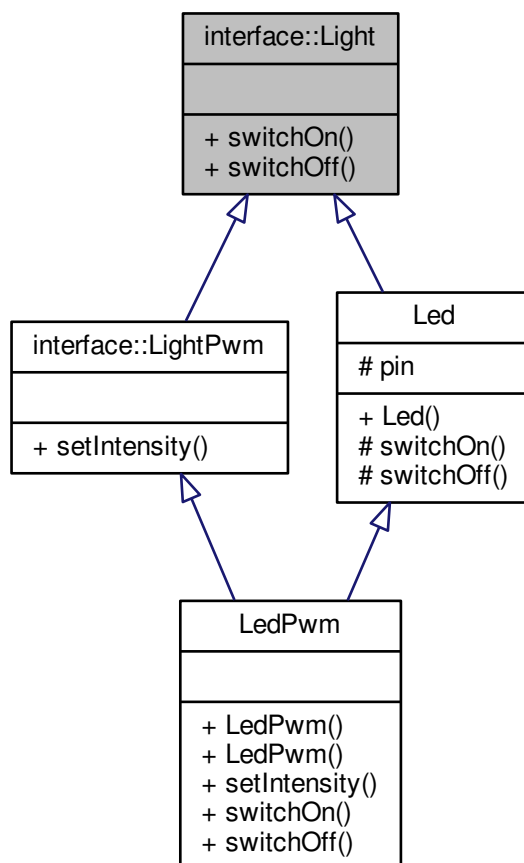
Class to manage the behaviour of a simple led

The documentation for this class was generated from the following files:
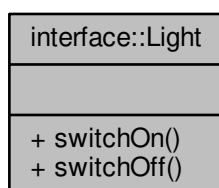
- src/task/LedTask.h
- src/task/LedTask.cpp

3.14   interface::Light Class Reference

`#include <Light.h>`

Inheritance diagram for interface::Light:

Collaboration diagram for interface::Light:

Public Member Functions

- virtual void **switchOn** ()=0
- virtual void **switchOff** ()=0

3.14.1 Detailed Description

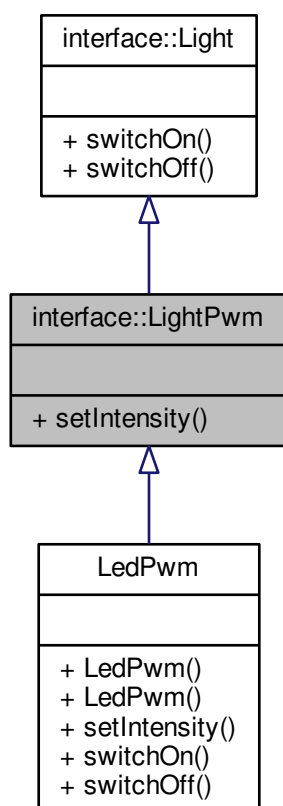Interface for devices that use light as a feedback

The documentation for this class was generated from the following file:
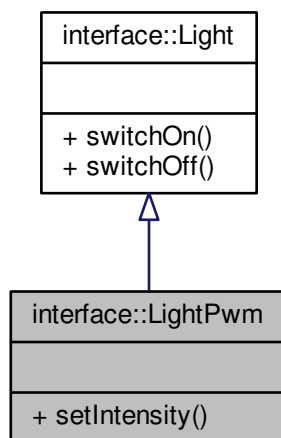
- src/interface/Light.h

3.15 interface::LightPwm Class Reference

```
#include <LightPwm.h>
```

Inheritance diagram for interface::LightPwm:

Collaboration diagram for interface::LightPwm:

Public Member Functions

- virtual void **setIntensity** (uint8_t)=0

3.15.1   Detailed Description

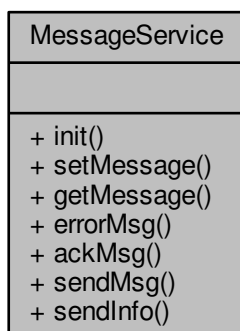Interface for devices that use a dimmable light as a feedback

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

- src/interface/LightPwm.h

3.16   MessageService Class Reference

```
#include <MessageService.h>
```

Collaboration diagram for MessageService:

```
┌─────────────────────────┐
│     MessageService      │
├─────────────────────────┤
│                         │
├─────────────────────────┤
│ + init()                │
│ + setMessage()          │
│ + getMessage()          │
│ + errorMsg()            │
│ + ackMsg()              │
│ + sendMsg()             │
│ + sendInfo()            │
└─────────────────────────┘
```

Public Member Functions

- void init (const int, const String &)

    Init the serial and print a welcome message
- void setMessage (String)

    Parse a message from the serial
- String **getMessage** ()
- void errorMsg ()

    Send a message error using JSON
- void ackMsg (const String)

    Send an ACK using JSON
- void sendMsg (const String, const String)

    Send a message using JSON
- void sendInfo (const int, const int, const uint8_t, const String)

    Send a service message using JSON

3.16.1 Detailed Description

Class to read and write data on serial

3.16.2 Member Function Documentation

3.16.2.1 void MessageService::init (const int *baud,* **const String &** *name* **)**

Init the serial and print a welcome message

Parameters

in	*baud*	The baud rate of the serial
in	*name*	The message to print

3.16.2.2 void MessageService::sendInfo (const int *distance,* const int *status,* const uint8_t *level,* const String *to*)

Send a service message using JSON

Create and send a JSON message with informations about the status of the game.

Parameters

in	*distance*	The distance read from the sonar
in	*status*	The status of the game or the lockpicking phase
in	*to*	The receiver of the message

3.16.2.3 void MessageService::sendMsg (const String *content,* const String *receiver*)

Send a message using JSON

Create and send a JSON message to a receiver.

Parameters

in	*content*	The content of the message
in	*receiver*	The receiver of the message

3.16.2.4 void MessageService::setMessage (String *msg*)

Parse a message from the serial

When a message is readed is parsed and if is valid reply with an ACK

Parameters

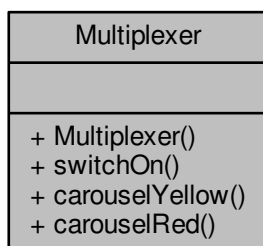| in	*msg*	The message received

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

- src/output/MessageService.h
- src/output/MessageService.cpp

3.17 Multiplexer Class Reference

```
#include <Multiplexer.h>
```

Collaboration diagram for Multiplexer:

Public Member Functions

- Multiplexer (int ∗, const int)

  Set the multiplex pins
- void switchOn (int)

  Select the output pin
- void carouselYellow (int)

  Run a 'carousel' effect on round led
- void **carouselRed** (int)

3.17.1 Detailed Description

Class to manage a 16-channels multiplexer (CD4067B)

3.17.2 Member Function Documentation

3.17.2.1 void Multiplexer::carouselYellow (int *del*)

Run a 'carousel' effect on round led

A round led is an array of 6 leds, set to HIGH each of the in sequence to create a 'carousel' effect. This led is binded with the multiplexer because Arduino Uno do not have enough pins.

Parameters

in	*del*	The delay time between each single shift

3.17.2.2 void Multiplexer::switchOn (int *output*)

Select the output pin

Select an output pin enabling the right channels using the truth table of the multiplexer

Parameters

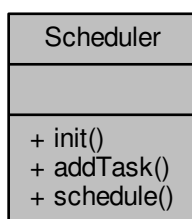| in	*output*	The pin to be enabled

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

- src/output/Multiplexer.h
- src/output/Multiplexer.cpp

3.18 Scheduler Class Reference

```
#include <Scheduler.h>
```

Collaboration diagram for Scheduler:

```
┌─────────────────┐
│    Scheduler    │
├─────────────────┤
│                 │
├─────────────────┤
│ + init()        │
│ + addTask()     │
│ + schedule()    │
└─────────────────┘
```

Public Member Functions

- void **init** (int)
- virtual bool **addTask** (Task ∗)
- virtual void **schedule** ()

3.18.1 Detailed Description

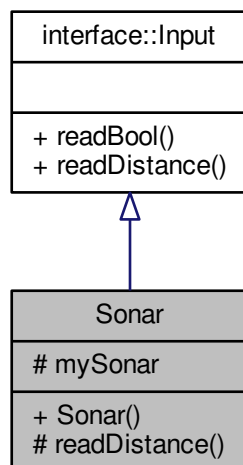Run with round robin scheduling all tasks

The documentation for this class was generated from the following files:
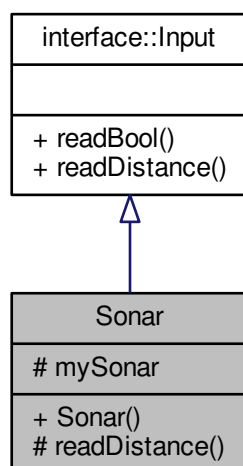
- src/control/Scheduler.h
- src/control/Scheduler.cpp

3.19 Sonar Class Reference

`#include <Sonar.h>`

Inheritance diagram for Sonar:

Collaboration diagram for Sonar:

Public Member Functions

- Sonar (int, int, int)

Create a new object to read value from sonar using 'NewPing' library

Protected Member Functions

- int readDistance ()

    Read the distance detected

Protected Attributes

- NewPing ∗ **mySonar**

3.19.1 Detailed Description

Class to manage the sonar (ultrasonic range detector)

3.19.2 Member Function Documentation

3.19.2.1 int Sonar::readDistance ( ) `[protected],[virtual]`

Read the distance detected

Returns

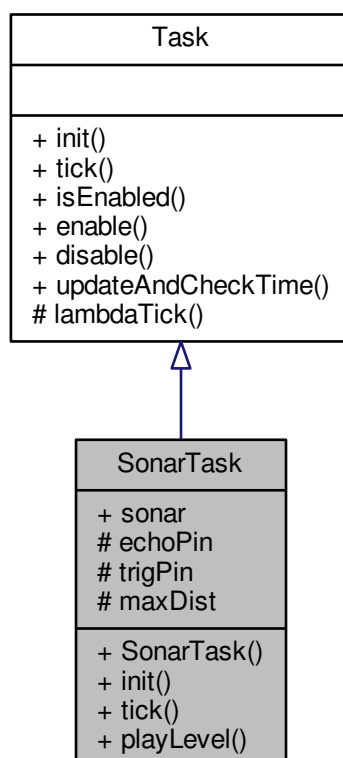The distance value in centimeters

Reimplemented from interface::Input.

The documentation for this class was generated from the following files:
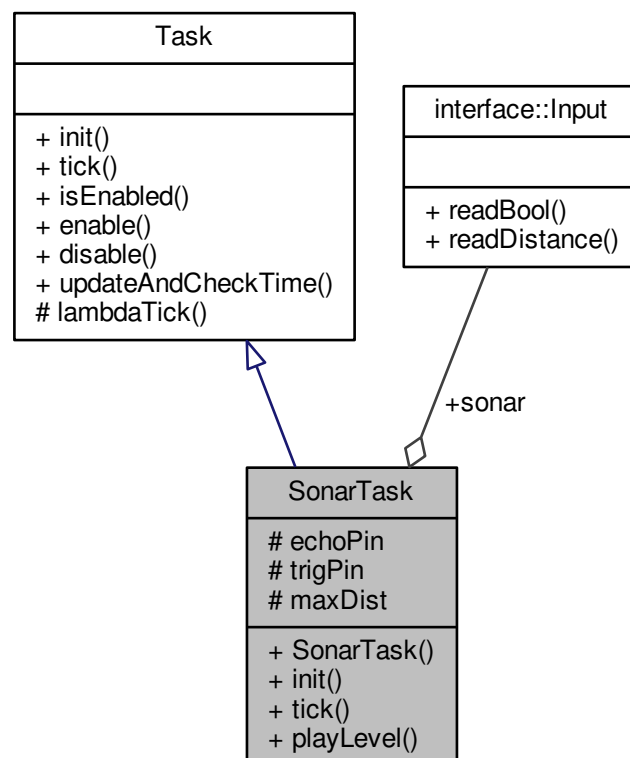
- src/input/Sonar.h
- src/input/Sonar.cpp

3.20 SonarTask Class Reference

```
#include <SonarTask.h>
```

Inheritance diagram for SonarTask:

```
┌─────────────────────────────────┐
│              Task               │
├─────────────────────────────────┤
│                                 │
├─────────────────────────────────┤
│ + init()                        │
│ + tick()                        │
│ + isEnabled()                   │
│ + enable()                      │
│ + disable()                     │
│ + updateAndCheckTime()          │
│ # lambdaTick()                  │
└─────────────────────────────────┘
                 △
                 │
┌─────────────────────────────────┐
│           SonarTask             │
├─────────────────────────────────┤
│ + sonar                         │
│ # echoPin                       │
│ # trigPin                       │
│ # maxDist                       │
├─────────────────────────────────┤
│ + SonarTask()                   │
│ + init()                        │
│ + tick()                        │
│ + playLevel()                   │
└─────────────────────────────────┘
```

Collaboration diagram for SonarTask:

Public Member Functions

- **SonarTask** (int, int, int, Context ∗)
- void **init** (int, void(∗)())
- void **tick** ()
- void **playLevel** ()

Data Fields

- interface::Input ∗ **sonar**

Protected Attributes

- int **echoPin**
- int **trigPin**
- int **maxDist**

Additional Inherited Members

3.20.1 Detailed Description
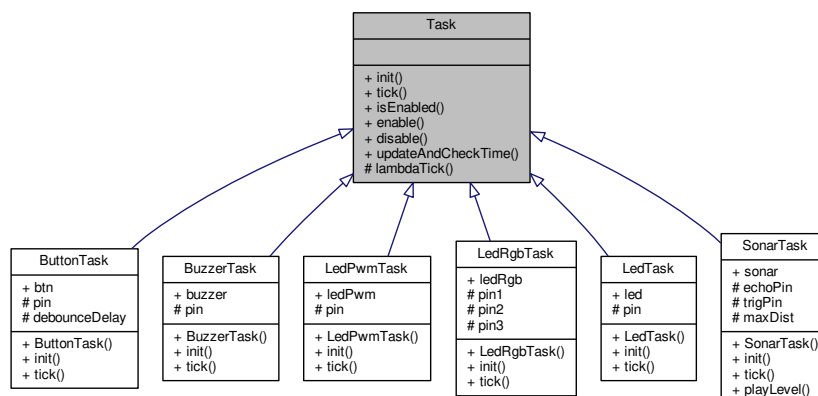
Class to manage the behaviour of the sonar

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

- src/task/SonarTask.h
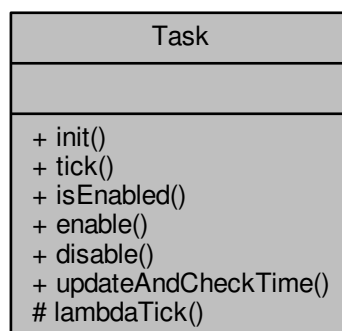- src/task/SonarTask.cpp

3.21 Task Class Reference

`#include <Task.h>`

Inheritance diagram for Task:

Collaboration diagram for Task:

Public Member Functions

- virtual void **init** (int period)
- virtual void **tick** ()=0
- bool **isEnabled** ()
- void **enable** ()
- void **disable** ()
- bool updateAndCheckTime (int basePeriod)

     Check if is time to do a context switch

Protected Member Functions

- virtual void lambdaTick ()=0

3.21.1   Detailed Description

Abstract class for tasks

3.21.2   Member Function Documentation

3.21.2.1   virtual void Task::lambdaTick ( ) `[protected],[pure virtual]`

The implementation of task

3.21.2.2   bool Task::updateAndCheckTime (int *basePeriod*) `[inline]`

Check if is time to do a context switch

Parameters

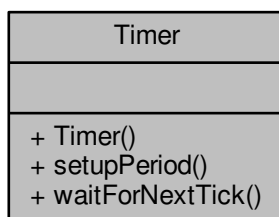| in	*basePeriod*	The period time of execution of the task

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

- src/task/Task.h

3.22   Timer Class Reference

```
#include <Timer.h>
```

Collaboration diagram for Timer:

```
┌─────────────────────────────┐
│            Timer            │
├─────────────────────────────┤
│                             │
├─────────────────────────────┤
│ + Timer()                   │
│ + setupPeriod()             │
│ + waitForNextTick()         │
└─────────────────────────────┘
```

Public Member Functions

- void **setupPeriod** (int)
- void **waitForNextTick** ()

3.22.1 Detailed Description

Class to manage Arduino internal timers

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

- src/control/Timer.h
- src/control/Timer.cpp