

Arduino Lib

1.0.0

Generated by Doxygen 1.8.14

Contents

1	arduino-avr	1
2	Class Index	5
2.1	Class List	5
3	File Index	7
3.1	File List	7
4	Class Documentation	9
4.1	DstSensor Struct Reference	9
4.1.1	Detailed Description	9
4.1.2	Member Data Documentation	9
4.1.2.1	echo	9
4.1.2.2	trigger	9
4.2	tuple2 Struct Reference	10
4.2.1	Member Data Documentation	10
4.2.1.1	a	10
4.2.1.2	b	10

5 File Documentation	11
5.1 <code>_delay.s</code> File Reference	11
5.2 <code>analog.c</code> File Reference	11
5.2.1 Macro Definition Documentation	11
5.2.1.1 <code>F_CPU</code>	12
5.2.1.2 <code>NULL</code>	12
5.2.2 Function Documentation	12
5.2.2.1 <code>AnalogWrite()</code>	12
5.2.2.2 <code>get_prescaler()</code>	12
5.2.2.3 <code>get_prescaler2()</code>	12
5.2.2.4 <code>sqr_wave()</code>	12
5.2.3 Variable Documentation	13
5.2.3.1 <code>_COM0</code>	13
5.2.3.2 <code>_COM1</code>	13
5.2.3.3 <code>_OCR</code>	13
5.2.3.4 <code>_TCCA</code>	13
5.2.3.5 <code>_TCCB</code>	13
5.3 <code>button.c</code> File Reference	13
5.3.1 Function Documentation	14
5.3.1.1 <code>Button_init()</code>	14
5.3.1.2 <code>Button_ispressed()</code>	14
5.3.1.3 <code>Button onclick()</code>	15
5.3.1.4 <code>Button_onrelease()</code>	15
5.4 <code>delay.c</code> File Reference	15
5.4.1 Function Documentation	16
5.4.1.1 <code>delay_1ms()</code>	16
5.4.1.2 <code>delay_1us()</code>	16
5.4.1.3 <code>delay_ms()</code>	16
5.4.1.4 <code>delay_us()</code>	16
5.5 <code>dst_sensor.c</code> File Reference	17

5.5.1	Function Documentation	17
5.5.1.1	DstSensor_init()	17
5.5.1.2	DstSensor_read()	17
5.6	include/analog.h File Reference	18
5.6.1	Function Documentation	18
5.6.1.1	AnalogWrite()	18
5.7	include/bit.h File Reference	18
5.7.1	Macro Definition Documentation	18
5.7.1.1	cbi	19
5.7.1.2	is_bit_set	19
5.7.1.3	sbi	19
5.8	include/button.h File Reference	19
5.8.1	Macro Definition Documentation	20
5.8.1.1	BUTTON_ONCLICK	20
5.8.1.2	BUTTON_ONRELEASE	20
5.8.2	Function Documentation	20
5.8.2.1	Button_init()	20
5.8.2.2	Button_ispressed()	21
5.8.2.3	Button_onclick()	21
5.8.2.4	Button_onrelease()	21
5.9	include/delay.h File Reference	22
5.9.1	Function Documentation	22
5.9.1.1	delay_ms()	22
5.9.1.2	delay_us()	22
5.10	include/dst_sensor.h File Reference	23
5.10.1	Function Documentation	23
5.10.1.1	DstSensor_init()	23
5.10.1.2	DstSensor_read()	24
5.11	include/led.h File Reference	24
5.11.1	Macro Definition Documentation	24

5.11.1.1	OFF	24
5.11.1.2	ON	25
5.11.2	Function Documentation	25
5.11.2.1	Led_blink()	25
5.11.2.2	Led_init()	25
5.11.2.3	Led_set()	25
5.11.2.4	Led_swap()	26
5.12	include/pin.h File Reference	26
5.12.1	Macro Definition Documentation	26
5.12.1.1	HIGH	27
5.12.1.2	INPUT	27
5.12.1.3	LOW	27
5.12.1.4	OUTPUT	27
5.12.1.5	PULL_UP	27
5.12.2	Function Documentation	27
5.12.2.1	DigitalRead()	27
5.12.2.2	DigitalWrite()	28
5.12.2.3	PinMode()	28
5.13	include/seg7.h File Reference	28
5.13.1	Typedef Documentation	29
5.13.1.1	seg7	29
5.13.2	Function Documentation	29
5.13.2.1	Seg7_init()	29
5.13.2.2	Seg7_putdigit()	29
5.14	include/sqr_wave.h File Reference	31
5.14.1	Function Documentation	31
5.14.1.1	sqr_wave()	31
5.15	led.c File Reference	31
5.15.1	Function Documentation	32
5.15.1.1	Led_blink()	32

5.15.1.2	Led_init()	32
5.15.1.3	Led_set()	32
5.15.1.4	Led_swap()	33
5.16	main.c File Reference	33
5.16.1	Function Documentation	33
5.16.1.1	main()	33
5.17	pin.c File Reference	34
5.17.1	Function Documentation	34
5.17.1.1	DigitalRead()	34
5.17.1.2	DigitalWrite()	34
5.17.1.3	PinMode()	35
5.17.2	Variable Documentation	35
5.17.2.1	_DDR	35
5.17.2.2	_PIN	35
5.17.2.3	_PORT	35
5.18	README.md File Reference	35
5.19	seg7.c File Reference	35
5.19.1	Enumeration Type Documentation	36
5.19.1.1	anonymous enum	36
5.19.2	Function Documentation	36
5.19.2.1	Seg7_init()	36
5.19.2.2	Seg7_putdigit()	38
Index		39

Chapter 1

arduino-avr

Helper functions for Arduino AVR

Documentação

Pinos

Os pinos são mapeados de maneira que possam ser utilizados a partir da numeração do Arduino. A manipulação dos pinos podem ser realizadas por meio das funções abaixo localizadas no arquivo [pin.h](#):

```
void PinMode(uint8_t pin, uint8_t mode);  
void DigitalWrite(uint8_t pin, uint8_t value);  
uint8_t DigitalRead(uint8_t pin);
```

Onde:

- Constantes para o modo do pino INPUT, OUTPUT ou PULL_UP.
- Constantes para o valor lógico do pino HIGH ou LOW.

Exemplo

```
#include "pin.h"  
  
#define LED 11  
  
int main(void) {  
  
    PinMode(LED, OUTPUT);  
    DigitalWrite(LED, HIGH);  
  
    while (1) {}  
  
    return 0;  
}
```

LED

Há um componente de LED localizado no arquivo `led.h`. A manipulação do LED pode ser feita através das seguintes funções:

```
void Led_init(uint8_t led);
void Led_set(uint8_t led, uint8_t state);
void Led_swap(uint8_t led);
void Led_blink(uint8_t led, uint32_t delay);
```

- A função `Led_init` é utilizada para configurar o pino como um LED.
- A função `Led_set` define o estado do LED, ON ou OFF.
- A função `Led_swap` troca o estado do LED, se está ON troca para OFF e se está OFF troca para ON.
- A função `Led_blink` recebe um parâmetro `delay` e pisca o LED por `delay` ms.

Exemplo

```
#include "led.h"

#define LED 11

int main(void) {
    Led_init(LED);

    while(1)
        Led_blink(LED, 500);

    return 0;
}
```

Atraso

Há duas funções de atraso, uma com precisão de milissegundos e outra com precisão de microsegundos. Ambas rotinas são implementadas em assembly no arquivo `_delay.s` e executadas a partir das funções definidas em `delay.h`.

```
void delay_ms(uint32_t ms);
void delay_us(uint32_t us);
```

Sensor de distância

O componente de sensor de distância é definido no arquivo `dst_sensor.h`. Ele consiste da definição de 2 pinos. Tal componente é manipulado pelas funções:

```
void DstSensor_init(DstSensor *dst);
uint32_t DstSensor_read(DstSensor *dst);
```

- A função `DstSensor_init` configura o componente.
- A função `DstSensor_read` realiza a leitura de distância e retorna o valor em cm.

A estrutura `DstSensor` é uma estrutura contendo os campos `trigger` e `echo`, que são pinos do componente.

Exemplo

```
#include "dst_sensor.h"

#define TRIGGER 9
#define ECHO 10

int main(void) {

    DstSensor dst = { .trigger = TRIGGER, .echo = ECHO };
    DstSensor_init(&dst);

    while (1) {
        uint32_t distance = DstSensor_read(&dst);
        // Use a distância lida.
    }

    return 0;
}
```

Display 7 segmentos

Button

```
#include "button.h"
#include "delay.h"
#include "led.h"
#include <stdbool.h>
#include <stddef.h>

void *on_click(void *args) {
    uint8_t led = *(uint8_t *)args;
    Led_swap(led);
    return NULL;
}

int main(void) {
    const uint8_t led = 2;
    const uint8_t btn = 3;

    Button_init(btn);
    Led_init(led);

    while (true) {
        Button_onclick(btn, on_click, (void *)&led);
        // ou
        // BUTTON_ONCLICK(btn, Led_swap(led));
    }

    return 0;
}
```


Chapter 2

Class Index

2.1 Class List

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

DstSensor	Estrutura que armazena os pinos necessários para manipulação do sensor de distância . . .	9
tuple2	10

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

_delay.s	11
analog.c	11
button.c	13
delay.c	15
dst_sensor.c	17
led.c	31
main.c	33
pin.c	34
seg7.c	35
include/analog.h	18
include/bit.h	18
include/button.h	19
include/delay.h	22
include/dst_sensor.h	23
include/led.h	24
include/pin.h	26
include/seg7.h	28
include/sqr_wave.h	31

Chapter 4

Class Documentation

4.1 DstSensor Struct Reference

Estrutura que armazena os pinos necessários para manipulação do sensor de distância.

```
#include <dst_sensor.h>
```

Public Attributes

- `uint8_t` [trigger](#)
- `uint8_t` [echo](#)

4.1.1 Detailed Description

Estrutura que armazena os pinos necessários para manipulação do sensor de distância.

4.1.2 Member Data Documentation

4.1.2.1 echo

```
uint8_t DstSensor::echo
```

4.1.2.2 trigger

```
uint8_t DstSensor::trigger
```

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

- `include/`[dst_sensor.h](#)

4.2 tuple2 Struct Reference

Public Attributes

- `uint16_t` [a](#)
- `uint8_t` [b](#)

4.2.1 Member Data Documentation

4.2.1.1 `a`

`uint16_t tuple2::a`

4.2.1.2 `b`

`uint8_t tuple2::b`

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

- [analog.c](#)

Chapter 5

File Documentation

5.1 `_delay.s` File Reference

5.2 `analog.c` File Reference

```
#include "analog.h"
#include "bit.h"
#include "pin.h"
#include "sqr_wave.h"
#include <avr/io.h>
```

Classes

- struct [tuple2](#)

Macros

- `#define NULL` (void *)0x0
- `#define F_CPU` 16000000UL

Functions

- void [AnalogWrite](#) (uint8_t pin, uint16_t dc)
Escreve uma onda PWM no pino com o duty cycle definido.
- [tuple2 get_prescaler](#) (double freq)
- [tuple2 get_prescaler2](#) (double freq)
- void [sqr_wave](#) (uint8_t pin, double freq)
Escreve uma onda quadrada no pino.

5.2.1 Macro Definition Documentation

5.2.1.1 F_CPU

```
#define F_CPU 16000000UL
```

5.2.1.2 NULL

```
#define NULL (void *)0x0
```

5.2 Function Documentation

5.2.2.1 AnalogWrite()

```
void AnalogWrite (
    uint8_t pin,
    uint16_t dc )
```

Escreve uma onda PWM no pino com o duty cycle definido.

Parameters

<i>pin</i>	Pino a escrever a onda, apenas os pinos 3, 5, 6, 9, 10 e 11 são suportados.
<i>dc</i>	Valor de duty cycle, o intervalo aceito é [0, 255]

5.2.2.2 get_prescaler()

```
tuple2 get_prescaler (
    double freq )
```

5.2.2.3 get_prescaler2()

```
tuple2 get_prescaler2 (
    double freq )
```

5.2.2.4 sqr_wave()

```
void sqr_wave (
    uint8_t pin,
    double freq )
```

Escreve uma onda quadrada no pino.

Parameters

<i>pin</i>	Numeração do pino. Pinos suportados 3, 5, 6, 9, 10 e 11
<i>freq</i>	Frequência a ser escrita, intervalo [30.51, 8000000]hz

5.2.3 Variable Documentation

5.2.3.1 _COM0

```
uint8_t _COM0
```

5.2.3.2 _COM1

```
uint8_t _COM1
```

5.2.3.3 _OCR

```
volatile uint16_t* _OCR
```

5.2.3.4 _TCCA

```
volatile uint8_t* _TCCA
```

5.2.3.5 _TCCB

```
volatile uint8_t* _TCCB
```

5.3 button.c File Reference

```
#include "button.h"  
#include <stddef.h>
```

Functions

- void `Button_init` (uint8_t pin)
Inicializa o pino como um botão.
- bool `Button_ispressed` (uint8_t pin)
Verifica se o botão está pressionado.
- void * `Button_onclick` (uint8_t pin, void *(*callback)(void *), void *args)
Observa um clique no botão e dispara o callback quando o clique acontecer.
- void * `Button_onrelease` (uint8_t pin, void *(*callback)(void *), void *args)
Observa um botão ser solto e dispara o callback.

5.3.1 Function Documentation

5.3.1.1 `Button_init()`

```
void Button_init (  
    uint8_t pin )
```

Inicializa o pino como um botão.

Parameters

<i>pin</i>	Numeração do pino a ser inicializado.
------------	---------------------------------------

5.3.1.2 `Button_ispressed()`

```
bool Button_ispressed (  
    uint8_t pin )
```

Verifica se o botão está pressionado.

Parameters

<i>pin</i>	Pino a realizar a verificação.
------------	--------------------------------

Returns

true Quando o pino está pressionado
false Quando o pino não está pressionado

5.3.1.3 Button_onclick()

```
void* Button_onclick (
    uint8_t pin,
    void (*)(void *) callback,
    void * args )
```

Observa um clique no botão e dispara o callback quando o clique acontecer.

Parameters

<i>pin</i>	Pino do botão a ser observado.
<i>callback</i>	Ponteiro para uma função que é executado quando o clique acontecer
<i>args</i>	Argumentos para o callback

Returns

void* retorna o valor retornado pelo callback.

5.3.1.4 Button_onrelease()

```
void* Button_onrelease (
    uint8_t pin,
    void (*)(void *) callback,
    void * args )
```

Observa um botão ser solto e dispara o callback.

Parameters

<i>pin</i>	Pino do botão a ser observado.
<i>callback</i>	Ponteiro para uma função que é executado quando o botão for solto acontecer
<i>args</i>	Argumentos para o callback

Returns

void* retorna o valor retornado pelo callback.

5.4 delay.c File Reference

```
#include "delay.h"
```

Functions

- void [delay_1us](#) (void)

- void `delay_1ms` (void)
- void `delay_ms` (uint32_t ms)
Realiza um atraso com precisão de milisegundos.
- void `delay_us` (uint32_t us)
Realiza um atraso com precisão de microsegundos.

5.4.1 Function Documentation

5.4.1.1 `delay_1ms()`

```
void delay_1ms (  
    void )
```

5.4.1.2 `delay_1us()`

```
void delay_1us (  
    void )
```

5.4.1.3 `delay_ms()`

```
void delay_ms (  
    uint32_t ms )
```

Realiza um atraso com precisão de milisegundos.

Parameters

<i>ms</i>	Quantidade de milisegundos para realizar o atraso.
-----------	--

5.4.1.4 `delay_us()`

```
void delay_us (  
    uint32_t us )
```

Realiza um atraso com precisão de microsegundos.

Parameters

<i>us</i>	Quantidade de microsegundos para realizar o atraso.
-----------	---

5.5 dst_sensor.c File Reference

```
#include "dst_sensor.h"
#include "bit.h"
#include "delay.h"
#include "pin.h"
#include <avr/io.h>
```

Functions

- void [DstSensor_init](#) ([DstSensor](#) *this)
Configura o sensor de distância.
- uint32_t [DstSensor_read](#) ([DstSensor](#) *this)
Realiza a leitura da distância.

5.5.1 Function Documentation

5.5.1.1 DstSensor_init()

```
void DstSensor_init (
    DstSensor * dst )
```

Configura o sensor de distância.

Ao chamar esta função, os pinos trigger e echo devem estar definidos.

Parameters

<i>dst</i>	Sensor de distância a ser configurado.
------------	--

5.5.1.2 DstSensor_read()

```
uint32_t DstSensor_read (
    DstSensor * dst )
```

Realiza a leitura da distância.

Parameters

<i>dst</i>	Sensor a ter a distância lida.
------------	--------------------------------

Returns

uint32_t Retorna a distância lida em cm.

5.6 include/analog.h File Reference

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

Functions

- void [AnalogWrite](#) (uint8_t pin, uint16_t dc)
Escreve uma onda PWM no pino com o duty cycle definido.

5.6.1 Function Documentation**5.6.1.1 AnalogWrite()**

```
void AnalogWrite (
    uint8_t pin,
    uint16_t dc )
```

Escreve uma onda PWM no pino com o duty cycle definido.

Parameters

<i>pin</i>	Pino a escrever a onda, apenas os pinos 3, 5, 6, 9, 10 e 11 são suportados.
<i>dc</i>	Valor de duty cycle, o intervalo aceito é [0, 255]

5.7 include/bit.h File Reference**Macros**

- #define [sbi](#)(REG, N) ((REG) |= (1 << (N)))
Ativa o bit N no registrador REG.
- #define [cbi](#)(REG, N) ((REG) &= ~(1 << (N)))
Desativa o bit N no registrador REG.
- #define [is_bit_set](#)(REG, N) (((REG) >> (N)) & 1)
Verifica se o bit N está ativo no registrador REG.

5.7.1 Macro Definition Documentation

5.7.1.1 cbi

```
#define cbi(  
    REG,  
    N ) ((REG) &= ~(1 << (N)))
```

Desativa o bit N no registrador REG.

5.7.1.2 is_bit_set

```
#define is_bit_set(  
    REG,  
    N ) (((REG) >> (N)) & 1)
```

Verifica se o bit N está ativo no registrador REG.

5.7.1.3 sbi

```
#define sbi(  
    REG,  
    N ) ((REG) |= (1 << (N)))
```

Ativa o bit N no registrador REG.

5.8 include/button.h File Reference

```
#include "delay.h"  
#include "pin.h"  
#include <stdbool.h>  
#include <stdint.h>
```

Macros

- `#define BUTTON_ONCLICK(btn, action)`
Macro que realiza uma ação quando o botão é pressionado.
- `#define BUTTON_ONRELEASE(btn, action)`
Macro que realiza uma ação quando o botão é solto.

Functions

- void `Button_init` (uint8_t pin)
Inicializa o pino como um botão.
- bool `Button_ispressed` (uint8_t pin)
Verifica se o botão está pressionado.
- void * `Button_onclick` (uint8_t pin, void (*)(callback)(void *), void *args)
Observa um clique no botão e dispara o callback quando o clique acontecer.
- void * `Button_onrelease` (uint8_t pin, void (*)(callback)(void *), void *args)
Observa um botão ser solto e dispara o callback.

5.8.1 Macro Definition Documentation

5.8.1.1 BUTTON_ONCLICK

```
#define BUTTON_ONCLICK(
    btn,
    action )
```

Value:

```
if (Button_ispressed(btn)) {
    action;
    while (Button_ispressed(btn))
        ;
    delay_ms(10);
}
```

Macro que realiza uma ação quando o botão é pressionado.

5.8.1.2 BUTTON_ONRELEASE

```
#define BUTTON_ONRELEASE(
    btn,
    action )
```

Value:

```
if (Button_ispressed(btn)) {
    while (Button_ispressed(btn))
        ;
    action;
    delay_ms(10);
}
```

Macro que realiza uma ação quando o botão é solto.

5.8.2 Function Documentation

5.8.2.1 Button_init()

```
void Button_init (
    uint8_t pin )
```

Inicializa o pino como um botão.

Parameters

<i>pin</i>	Numeração do pino a ser inicializado.
------------	---------------------------------------

5.8.2.2 Button_ispressed()

```
bool Button_ispressed (
    uint8_t pin )
```

Verifica se o botão está pressionado.

Parameters

<i>pin</i>	Pino a realizar a verificação.
------------	--------------------------------

Returns

true Quando o pino está pressionado
false Quando o pino não está pressionado

5.8.2.3 Button_onclick()

```
void* Button_onclick (
    uint8_t pin,
    void (*)(void *) callback,
    void * args )
```

Observa um clique no botão e dispara o callback quando o clique acontecer.

Parameters

<i>pin</i>	Pino do botão a ser observado.
<i>callback</i>	Ponteiro para uma função que é executado quando o clique acontecer
<i>args</i>	Argumentos para o callback

Returns

void* retorna o valor retornado pelo callback.

5.8.2.4 Button_onrelease()

```
void* Button_onrelease (
    uint8_t pin,
```

```
void (*)(void *) callback,
void * args )
```

Observa um botão ser solto e dispara o callback.

Parameters

<i>pin</i>	Pino do botão a ser observado.
<i>callback</i>	Ponteiro para uma função que é executado quando o botão for solto acontecer
<i>args</i>	Argumentos para o callback

Returns

void* retorna o valor retornado pelo callback.

5.9 include/delay.h File Reference

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

Functions

- void [delay_ms](#) (uint32_t ms)
Realiza um atraso com precisão de milisegundos.
- void [delay_us](#) (uint32_t us)
Realiza um atraso com precisão de microsegundos.

5.9.1 Function Documentation

5.9.1.1 delay_ms()

```
void delay_ms (
    uint32_t ms )
```

Realiza um atraso com precisão de milisegundos.

Parameters

<i>ms</i>	Quantidade de milisegundos para realizar o atraso.
-----------	--

5.9.1.2 delay_us()

```
void delay_us (
```

```
uint32_t us )
```

Realiza um atraso com precisão de microsegundos.

Parameters

<i>us</i>	Quantidade de microsegundos para realizar o atraso.
-----------	---

5.10 include/dst_sensor.h File Reference

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

Classes

- struct [DstSensor](#)

Estrutura que armazena os pinos necessários para manipulação do sensor de distância.

Functions

- void [DstSensor_init](#) ([DstSensor](#) *dst)
Configura o sensor de distância.
- uint32_t [DstSensor_read](#) ([DstSensor](#) *dst)
Realiza a leitura da distância.

5.10.1 Function Documentation

5.10.1.1 DstSensor_init()

```
void DstSensor_init (  
    DstSensor * dst )
```

Configura o sensor de distância.

Ao chamar esta função, os pinos trigger e echo devem estar definidos.

Parameters

<i>dst</i>	Sensor de distância a ser configurado.
------------	--

5.10.1.2 DstSensor_read()

```
uint32_t DstSensor_read (
    DstSensor * dst )
```

Realiza a leitura da distância.

Parameters

<i>dst</i>	Sensor a ter a distância lida.
------------	--------------------------------

Returns

uint32_t Retorna a distância lida em cm.

5.11 include/led.h File Reference

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

Macros

- #define **OFF** 0x0
Constante que define o estado desligado do LED.
- #define **ON** 0x1
Constante que define o estado ligado do LED.

Functions

- void **Led_init** (uint8_t led)
Inicializa o pino como um LED.
- void **Led_set** (uint8_t led, uint8_t state)
Atribui o estado ON ou OFF ao LED.
- void **Led_swap** (uint8_t led)
Troca o estado do LED.
- void **Led_blink** (uint8_t led, uint32_t duration)
Pisca o LED com uma duração.

5.11.1 Macro Definition Documentation

5.11.1.1 OFF

```
#define OFF 0x0
```

Constante que define o estado desligado do LED.

5.11.1.2 ON

```
#define ON 0x1
```

Constante que define o estado ligado do LED.

5.11.2 Function Documentation

5.11.2.1 Led_blink()

```
void Led_blink (
    uint8_t led,
    uint32_t duration )
```

Pisca o LED com uma duração.

Parameters

<i>led</i>	Pino do LED
<i>duration</i>	Duração em milisegundos.

5.11.2.2 Led_init()

```
void Led_init (
    uint8_t led )
```

Inicializa o pino como um LED.

Parameters

<i>led</i>	Pino a ser utilizado como LED
------------	-------------------------------

5.11.2.3 Led_set()

```
void Led_set (
    uint8_t led,
    uint8_t state )
```

Atribui o estado ON ou OFF ao LED.

Parameters

<i>led</i>	Pino a ser atribuído o estado.
<i>state</i>	Estado ON ou OFF para atribuir ao pino.

5.11.2.4 Led_swap()

```
void Led_swap (
    uint8_t led )
```

Troca o estado do LED.

Parameters

<i>led</i>	Pino a ter o estado trocado.
------------	------------------------------

5.12 include/pin.h File Reference

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

Macros

- `#define INPUT 0x0`
- `#define OUTPUT 0x1`
- `#define PULL_UP 0x2`
- `#define LOW 0x0`
- `#define HIGH 0x1`

Functions

- void `PinMode` (uint8_t pin, uint8_t mode)
Define um pino como entrada, saída ou pull up.
- void `DigitalWrite` (uint8_t pin, uint8_t value)
Escreve o valor lógico 1 ou 0 no pino especificado.
- uint8_t `DigitalRead` (uint8_t pin)
Lê o valor lógico do pino.

5.12.1 Macro Definition Documentation

5.12.1.1 HIGH

```
#define HIGH 0x1
```

5.12.1.2 INPUT

```
#define INPUT 0x0
```

5.12.1.3 LOW

```
#define LOW 0x0
```

5.12.1.4 OUTPUT

```
#define OUTPUT 0x1
```

5.12.1.5 PULL_UP

```
#define PULL_UP 0x2
```

5.12.2 Function Documentation

5.12.2.1 DigitalRead()

```
uint8_t DigitalRead (  
    uint8_t pin ) [inline]
```

Lê o valor lógico do pino.

Parameters

<i>pin</i>	Número do pino a ser lido.
------------	----------------------------

Returns

uint8_t Retorna o valor lógico lido, podendo ser HIGH ou LOW.

5.12.2.2 DigitalWrite()

```
void DigitalWrite (
    uint8_t pin,
    uint8_t value ) [inline]
```

Escreve o valor lógico 1 ou 0 no pino especificado.

Parameters

<i>pin</i>	Número do pino a ser escrito.
<i>value</i>	Valor lógico a ser escrito no pino, podendo ser HIGH ou LOW.

5.12.2.3 PinMode()

```
void PinMode (
    uint8_t pin,
    uint8_t mode ) [inline]
```

Define um pino como entrada, saída ou pull up.

Parameters

<i>pin</i>	Número do pino a ser configurado.
<i>mode</i>	Modo de configuração do pino, podendo ser INPUT, OUTPUT ou PULL_UP

5.13 include/seg7.h File Reference

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

Typedefs

- typedef uint8_t [seg7](#)[8]
Display de 7 segmentos, contam todo os 7 pinos mais o ponto.

Functions

- void [Seg7_init](#) ([seg7](#) display, uint8_t a, uint8_t b, uint8_t c, uint8_t d, uint8_t e, uint8_t f, uint8_t g, uint8_t dp)
Inicializa o display com os pinos.
- void [Seg7_putdigit](#) ([seg7](#) display, uint8_t digit)
Escreve um dígito no display.

5.13.1 Typedef Documentation

5.13.1.1 seg7

```
typedef uint8_t seg7[8]
```

Display de 7 segmentos, contem todo os 7 pinos mais o ponto.

5.13.2 Function Documentation

5.13.2.1 Seg7_init()

```
void Seg7_init (
    seg7 display,
    uint8_t a,
    uint8_t b,
    uint8_t c,
    uint8_t d,
    uint8_t e,
    uint8_t f,
    uint8_t g,
    uint8_t dp )
```

Inicializa o display com os pinos.

Parameters

<i>display</i>	Componente do display a ser configurado.
<i>a</i>	Pino A
<i>b</i>	Pino B
<i>c</i>	Pino C
<i>d</i>	Pino D
<i>e</i>	Pino E
<i>f</i>	Pino F
<i>g</i>	Pino G
<i>dp</i>	Pino DP

5.13.2.2 Seg7_putdigit()

```
void Seg7_putdigit (
    seg7 display,
    uint8_t digit )
```

Escreve um dígito no display.

Parameters

<i>display</i>	Componente de display a ser escrito
<i>digit</i>	Digito que deve ser escrito

5.14 include/sqr_wave.h File Reference

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

Functions

- void [sqr_wave](#) (uint8_t pin, double freq)
Escreve uma onda quadrada no pino.

5.14.1 Function Documentation

5.14.1.1 [sqr_wave\(\)](#)

```
void sqr_wave (  
    uint8_t pin,  
    double freq )
```

Escreve uma onda quadrada no pino.

Parameters

<i>pin</i>	Numeração do pino. Pinos suportados 3, 5, 6, 9, 10 e 11
<i>freq</i>	Frequência a ser escrita, intervalo [30.51, 8000000]hz

5.15 led.c File Reference

```
#include "led.h"  
#include "delay.h"
```

Functions

- void [Led_init](#) (uint8_t led)
Inicializa o pino como um LED.
- void [Led_set](#) (uint8_t led, uint8_t state)

Atribui o estado ON ou OFF ao LED.

- void [Led_swap](#) (uint8_t led)

Troca o estado do LED.

- void [Led_blink](#) (uint8_t led, uint32_t duration)

Pisca o LED com uma duração.

5.15.1 Function Documentation

5.15.1.1 Led_blink()

```
void Led_blink (
    uint8_t led,
    uint32_t duration )
```

Pisca o LED com uma duração.

Parameters

<i>led</i>	Pino do LED
<i>duration</i>	Duração em milisegundos.

5.15.1.2 Led_init()

```
void Led_init (
    uint8_t led )
```

Inicializa o pino como um LED.

Parameters

<i>led</i>	Pino a ser utilizado como LED
------------	-------------------------------

5.15.1.3 Led_set()

```
void Led_set (
    uint8_t led,
    uint8_t state )
```

Atribui o estado ON ou OFF ao LED.

Parameters

<i>led</i>	Pino a ser atribuído o estado.
<i>state</i>	Estado ON ou OFF para atribuir ao pino.

5.15.1.4 Led_swap()

```
void Led_swap (
    uint8_t led )
```

Troca o estado do LED.

Parameters

<i>led</i>	Pino a ter o estado trocado.
------------	------------------------------

5.16 main.c File Reference

```
#include "delay.h"
#include "analog.h"
#include "bit.h"
#include "dst_sensor.h"
#include "led.h"
#include "sqr_wave.h"
#include "uart.h"
#include <avr/io.h>
#include <math.h>
#include <stdbool.h>
```

Functions

- int [main](#) (void)

5.16.1 Function Documentation

5.16.1.1 main()

```
int main (
    void )
```

5.17 pin.c File Reference

```
#include <avr/io.h>
#include <bit.h>
#include <pin.h>
#include <stdio.h>
```

Functions

- void [PinMode](#) (uint8_t pin, uint8_t mode)
Define um pino como entrada, saída ou pull up.
- void [DigitalWrite](#) (uint8_t pin, uint8_t value)
Escreve o valor lógico 1 ou 0 no pino especificado.
- uint8_t [DigitalRead](#) (uint8_t pin)
Lê o valor lógico do pino.

5.17.1 Function Documentation

5.17.1.1 [DigitalRead\(\)](#)

```
uint8_t DigitalRead (
    uint8_t pin ) [inline]
```

Lê o valor lógico do pino.

Parameters

<i>pin</i>	Número do pino a ser lido.
------------	----------------------------

Returns

uint8_t Retorna o valor lógico lido, podendo ser HIGH ou LOW.

5.17.1.2 [DigitalWrite\(\)](#)

```
void DigitalWrite (
    uint8_t pin,
    uint8_t value ) [inline]
```

Escreve o valor lógico 1 ou 0 no pino especificado.

Parameters

<i>pin</i>	Número do pino a ser escrito.
<i>value</i>	Valor lógico a ser escrito no pino, podendo ser HIGH ou LOW.

5.17.1.3 PinMode()

```
void PinMode (
    uint8_t pin,
    uint8_t mode ) [inline]
```

Define um pino como entrada, saída ou pull up.

Parameters

<i>pin</i>	Número do pino a ser configurado.
<i>mode</i>	Modo de configuração do pino, podendo ser INPUT, OUTPUT ou PULL_UP

5.17.2 Variable Documentation

5.17.2.1 _DDR

```
volatile uint8_t* _DDR
```

5.17.2.2 _PIN

```
volatile uint8_t* _PIN
```

5.17.2.3 _PORT

```
volatile uint8_t* _PORT
```

5.18 README.md File Reference

5.19 seg7.c File Reference

```
#include "seg7.h"
#include "pin.h"
```

Enumerations

- enum {
 A = 0, B, C, D,
 E, F, G, DP }

Functions

- void [Seg7_init](#) ([seg7](#) display, uint8_t a, uint8_t b, uint8_t c, uint8_t d, uint8_t e, uint8_t f, uint8_t g, uint8_t dp)
Inicializa o display com os pinos.
- void [Seg7_putdigit](#) ([seg7](#) display, uint8_t digit)
Escreve um dígito no display.

5.19.1 Enumeration Type Documentation

5.19.1.1 anonymous enum

anonymous enum

Enumerator

A	
B	
C	
D	
E	
F	
G	
DP	

5.19.2 Function Documentation

5.19.2.1 Seg7_init()

```
void Seg7_init (  
    seg7 display,  
    uint8_t a,  
    uint8_t b,  
    uint8_t c,  
    uint8_t d,  
    uint8_t e,  
    uint8_t f,
```

```
uint8_t g,  
uint8_t dp )
```

Inicializa o display com os pinos.

Parameters

<i>display</i>	Componente do display a ser configurado.
<i>a</i>	Pino A
<i>b</i>	Pino B
<i>c</i>	Pino C
<i>d</i>	Pino D
<i>e</i>	Pino E
<i>f</i>	Pino F
<i>g</i>	Pino G
<i>dp</i>	Pino DP

5.19.2.2 Seg7_putdigit()

```
void Seg7_putdigit (
    seg7 display,
    uint8_t digit )
```

Escreve um dígito no display.

Parameters

<i>display</i>	Componente de display a ser escrito
<i>digit</i>	Dígito que deve ser escrito

Index

- `_COM0`
 - `analog.c`, 13
- `_COM1`
 - `analog.c`, 13
- `_DDR`
 - `pin.c`, 35
- `_OCR`
 - `analog.c`, 13
- `_PIN`
 - `pin.c`, 35
- `_PORT`
 - `pin.c`, 35
- `_TCCA`
 - `analog.c`, 13
- `_TCCB`
 - `analog.c`, 13
- `_delay.s`, 11

- `a`
 - `tuple2`, 10
- `analog.c`, 11
 - `_COM0`, 13
 - `_COM1`, 13
 - `_OCR`, 13
 - `_TCCA`, 13
 - `_TCCB`, 13
 - `AnalogWrite`, 12
 - `F_CPU`, 11
 - `get_prescaler`, 12
 - `get_prescaler2`, 12
 - `NULL`, 12
 - `sqr_wave`, 12
- `analog.h`
 - `AnalogWrite`, 18
- `AnalogWrite`
 - `analog.c`, 12
 - `analog.h`, 18

- `b`
 - `tuple2`, 10
- `BUTTON_ONCLICK`
 - `button.h`, 20
- `BUTTON_ONRELEASE`
 - `button.h`, 20
- `bit.h`
 - `cbi`, 18
 - `is_bit_set`, 19
 - `sbi`, 19
- `button.c`, 13
 - `Button_init`, 14

- `Button_ispressed`, 14
- `Button_onclick`, 14
- `Button_onrelease`, 15
- `button.h`
 - `BUTTON_ONCLICK`, 20
 - `BUTTON_ONRELEASE`, 20
 - `Button_init`, 20
 - `Button_ispressed`, 21
 - `Button_onclick`, 21
 - `Button_onrelease`, 21
- `Button_init`
 - `button.c`, 14
 - `button.h`, 20
- `Button_ispressed`
 - `button.c`, 14
 - `button.h`, 21
- `Button_onclick`
 - `button.c`, 14
 - `button.h`, 21
- `Button_onrelease`
 - `button.c`, 15
 - `button.h`, 21
- `cbi`
 - `bit.h`, 18
- `delay.c`, 15
 - `delay_1ms`, 16
 - `delay_1us`, 16
 - `delay_ms`, 16
 - `delay_us`, 16
- `delay.h`
 - `delay_ms`, 22
 - `delay_us`, 22
- `delay_1ms`
 - `delay.c`, 16
- `delay_1us`
 - `delay.c`, 16
- `delay_ms`
 - `delay.c`, 16
 - `delay.h`, 22
- `delay_us`
 - `delay.c`, 16
 - `delay.h`, 22
- `DigitalRead`
 - `pin.c`, 34
 - `pin.h`, 27
- `DigitalWrite`
 - `pin.c`, 34
 - `pin.h`, 28

- dst_sensor.c, [17](#)
 - DstSensor_init, [17](#)
 - DstSensor_read, [17](#)
- dst_sensor.h
 - DstSensor_init, [23](#)
 - DstSensor_read, [23](#)
- DstSensor, [9](#)
 - echo, [9](#)
 - trigger, [9](#)
- DstSensor_init
 - dst_sensor.c, [17](#)
 - dst_sensor.h, [23](#)
- DstSensor_read
 - dst_sensor.c, [17](#)
 - dst_sensor.h, [23](#)
- echo
 - DstSensor, [9](#)
- F_CPU
 - analog.c, [11](#)
- get_prescaler
 - analog.c, [12](#)
- get_prescaler2
 - analog.c, [12](#)
- HIGH
 - pin.h, [26](#)
- INPUT
 - pin.h, [27](#)
- include/analog.h, [18](#)
- include/bit.h, [18](#)
- include/button.h, [19](#)
- include/delay.h, [22](#)
- include/dst_sensor.h, [23](#)
- include/led.h, [24](#)
- include/pin.h, [26](#)
- include/seg7.h, [28](#)
- include/sqr_wave.h, [31](#)
- is_bit_set
 - bit.h, [19](#)
- LOW
 - pin.h, [27](#)
- led.c, [31](#)
 - Led_blink, [32](#)
 - Led_init, [32](#)
 - Led_set, [32](#)
 - Led_swap, [33](#)
- led.h
 - Led_blink, [25](#)
 - Led_init, [25](#)
 - Led_set, [25](#)
 - Led_swap, [26](#)
 - OFF, [24](#)
 - ON, [24](#)
- Led_blink
 - led.c, [32](#)
 - led.h, [25](#)
- Led_init
 - led.c, [32](#)
 - led.h, [25](#)
- Led_set
 - led.c, [32](#)
 - led.h, [25](#)
- Led_swap
 - led.c, [33](#)
 - led.h, [26](#)
- main
 - main.c, [33](#)
- main.c, [33](#)
 - main, [33](#)
- NULL
 - analog.c, [12](#)
- OFF
 - led.h, [24](#)
- OUTPUT
 - pin.h, [27](#)
- ON
 - led.h, [24](#)
- PULL_UP
 - pin.h, [27](#)
- pin.c, [34](#)
 - _DDR, [35](#)
 - _PIN, [35](#)
 - _PORT, [35](#)
 - DigitalRead, [34](#)
 - DigitalWrite, [34](#)
 - PinMode, [35](#)
- pin.h
 - DigitalRead, [27](#)
 - DigitalWrite, [28](#)
 - HIGH, [26](#)
 - INPUT, [27](#)
 - LOW, [27](#)
 - OUTPUT, [27](#)
 - PULL_UP, [27](#)
 - PinMode, [28](#)
- PinMode
 - pin.c, [35](#)
 - pin.h, [28](#)
- README.md, [35](#)
- sbi
 - bit.h, [19](#)
- seg7
 - seg7.h, [29](#)
- seg7.c, [35](#)
 - Seg7_init, [36](#)
 - Seg7_putdigit, [38](#)
- seg7.h
 - seg7, [29](#)
 - Seg7_init, [29](#)

- Seg7_putdigit, [29](#)
- Seg7_init
 - seg7.c, [36](#)
 - seg7.h, [29](#)
- Seg7_putdigit
 - seg7.c, [38](#)
 - seg7.h, [29](#)
- sqr_wave
 - analog.c, [12](#)
 - sqr_wave.h, [31](#)
- sqr_wave.h
 - sqr_wave, [31](#)
- trigger
 - DstSensor, [9](#)
- tuple2, [10](#)
 - a, [10](#)
 - b, [10](#)