Dimmer Documentation

Release 1.0

Dimmer

July 26, 2016

Contents

This is an Arduino software library to control AC loads using triacs and a zero cross detector circuit. The library methods can be used to control the AC load power for multiple triacs independently, using a single shared zero-cross circuit.

- Source code: https://github.com/circuitar/Dimmer
- Documentation: http://dimmer.readthedocs.org/
- Reference Board: Triac Nanoshield and Zero Cross Nanoshield from Circuitar

There are different ways to implement zero cross detector circuits. This library is based on the implementation above, but it can be easily adapted to use any type of zero cross detector circuit.

To install, just click **Download ZIP** and install it using **Sketch > Include Library... > Add .ZIP Library** in the Arduino IDE.

The following examples are provided:

- Fade Arduino Fade example using an AC lamp.
- FadeMinimum Arduino Fade example using an AC lamp and setting a minimum power level (useful for dimmable LED or CFL lamps).
- RandomLamps Control 3 dimmable lamps with random values (can be extended to 10 lamps).
- WaveLamps Control 3 dimmable lamps in a wave form (can be extended to 10 lamps).
- CountMode Control high, low response AC loads without introducing noise using count mode.

CHAPTER 1

Class Documentation

class `Dimmer`

A dimmer channel.

This object can control the power delivered to an AC load using a triac and a zero cross circuit. A common zero cross circuit is shared across all instances, but each instance controls a separate triac.

Public Functions

`Dimmer` (uint8_t *pin*, uint8_t *mode* = DIMMER_NORMAL, double *rampTime* = 1.5, uint8_t *freq* = 60)

Constructor.

See rampTime COUNT_MODE: Counts AC waves and applies full half cycles from time to time.

See *setRampTime()*.

See *begin()*

Parameters

- `pin` - pin that activates the triac.
- `mode` - operating mode. Possible modes: NORMAL_MODE: Uses timer to apply only a percentage of the AC power to the lamp every half cycle. RAMP_MODE: Same as in normal mode, but it applies a ramp effect when changing levels.

Parameters

- `rampTime` - time it takes for the value to rise from 0% to 100% in RAMP_MODE, in seconds. Default 1.5.

Parameters

- `freq` - AC frequency, in Hz. Supported values are 60Hz and 50Hz, use others at your own risk.

void **`begin`** (uint8_t *value* = 0, bool *on* = true)

Initializes the module.

Initializes zero cross circuit and Timer 2 interrupts. Set the lamp state and value according to initial settings.

Parameters

- `value` - initial intensity of the lamp, as a percentage. Minimum is 0, maximum is 100 and default is 0.

- `on` - initial lamp state. True if lamp is on or false if it's off. Lamp is on by default.

void **`off`**()
Turns the lamp OFF.

void **`on`**()
Turns the lamp ON.

void **`toggle`**()
Toggles the lamp on/off state.

uint8_t **`getValue`**()
Gets the current value (intensity) of the lamp.

Return current lamp value, from 0 to 100.

bool **`getState`**()
Gets the current state of the lamp.

Return current lamp state. ON or OFF.

void **`set`**(uint8_t *value*)
Sets the value of the lamp.

Parameters

- `value` - the value (intensity) of the lamp. Accepts values from 0 to 100.

void **`set`**(uint8_t *value*, bool *on*)
Sets the value and the state of the lamp.

Parameters

- `value` - value (intensity) of the lamp. Accepts values from 0 to 100.
- `on` - state of the lamp. True turns lamp on, false turns lamp off.

void **`setMinimum`**(uint8_t *value*)
Sets the mimimum acceptable power level.

This is useful to control loads that cannot be dimmed to a very low level, like dimmable LED or CFL lamps.

Parameters

- `value` - the minimum value (intensity) to use. Accepts values from 0 to 100.

void **`setRampTime`**(double *rampTime*)
Sets tje time it takes for the value to rise from 0% to 100% in RAMP_MODE, in seconds.

Parameters

- `value` - the ramp time. Maximum is 2^16 / (2 * AC frequency)

This documentation was built using ArduinoDocs.

D