```
/*
 * Defines the functionality for the WS2811LED class.
 *
 * Version 0.0   2022.12.02 LWH created
 *
 * Copyright (c) 2022, LWH brainware. All rights reserved.
 */
#define DEBUG         0

#include <LWHlog.h>
#include "WS2811LED.h"

/*****************************************************************
 *   Creates a new WS2811LED instance.
 *   creator must add it to the proper index
 */
WS2811LED::WS2811LED() {
  myidx       = 0;
  theState    = LEDState::OFF;
  theObserver = nullptr;
  currentbrightness = 0;
  nominalbrightness = 128;
  minimumBrightness = 0;
};

/*****************************************************************
 * sets the index from the chain and its index in the chain
 */
void WS2811LED::setIdx(uint8_t  achn, uint8_t  aidx) {
  myidx = achn*100 + aidx;
}

/*****************************************************************
 *   sets the observer to notify when the brightness has changed.
 *   set to null to unregister
 */
void WS2811LED::setObserver(LEDObserver* aobserver)  {
  theObserver = aobserver;
};

/*****************************************************************
 */
void WS2811LED::loop(void) {
  switch (theState) {
  case LEDState::TRANSIENT_ON:
    animateOn();
    break;
  case LEDState::ON:
    animateLoop();
    break;
  case LEDState::TRANSIENT_OFF:
    animateOff();
    break;
  default:
    break;
  }
};

/*****************************************************************
 * returns the current LED brightness.
 * (for simple LEDs the same as the nominal brightness)
 */
uint8_t  WS2811LED::getCurrentBrightness() {
  return currentbrightness;
};

/*****************************************************************
 * Sets the current LED brightness.
 * (for simple LEDs the same as the nominal brightness)
 * notifies any listeners that the brightness has changed
 */
```

```
void WS2811LED::setCurrentBrightness(uint8_t abrite) {
  debug("LED %03i cb %i\n", myidx, abrite);
  currentbrightness  = abrite;
  if (theObserver != nullptr) {
    theObserver->updateFromLED(false);
  }
};

/**************************************************************************
 * returns the nominal/maximum LED brightness.
 */
uint8_t WS2811LED::getNominalBrightness() {
  return nominalbrightness;
};

/**************************************************************************
 * Sets the new LED nominal/maximum brightness.
 * if on, changes also current brightness
 */
void WS2811LED::setNominalBrightness(uint8_t abrite) {
  debug("LED %03i nb %i\n", myidx, abrite);
  nominalbrightness  = abrite;
  if (theState == LEDState::ON) {
    setCurrentBrightness(abrite);
  }
};

/**************************************************************************
 * returns the minimum LED brightness.
 */
uint8_t WS2811LED::getMinimumBrightness() {
  return minimumBrightness;
};

/**************************************************************************
 * Sets the new LED minimum brightness.
 * for animations only.
 */
void WS2811LED::setMinimumBrightness(uint8_t abrite) {
  debug("LED %03i mb %i\n", myidx, abrite);
  minimumBrightness  = abrite;
};

/**************************************************************************
 * switches the LED on.
 * performs any transitions (fade, flicker etc.)
 */
void  WS2811LED::setOn(bool animate)  {
  if (theState == LEDState::ON) {
    return;
  }
  if (animate) {
    setCurrentBrightness(minimumBrightness);
    setState(LEDState::TRANSIENT_ON);
  } else {
    setCurrentBrightness(nominalbrightness);
    setState(LEDState::ON);
  }
};

/**************************************************************************
 * switches the LED off.
 * performs any transitions (fade, flicker etc.)
 */
void  WS2811LED::setOff(bool animate) {
  if (theState == LEDState::OFF) {
    return;
  }
  if (animate) {
    setState(LEDState::TRANSIENT_OFF);
  } else {
```

```
    setCurrentBrightness(0);
    setState(LEDState::OFF);
  }
};

/*****************************************************************
 * perform the transition when the LED is switched on
 */
void  WS2811LED::animateOn() {
  setCurrentBrightness(nominalbrightness);
  setState(LEDState::ON);
};

/*****************************************************************
 * perform something while the LED is on
 */
void  WS2811LED::animateLoop()  {
  // nothing to do here
};

/*****************************************************************
 * perform the transition when the LED is switched off
 */
void  WS2811LED::animateOff()  {
  setCurrentBrightness(0);
  setState(LEDState::OFF);
};

/*****************************************************************
 * set the led state
 */
void  WS2811LED::setState(LEDState  astate)  {
  debug("LED %03i st %i\n", myidx, (int) astate);
  theState  = astate;
}

/*****************************************************************
 * returns the current led state
 */
LEDState   WS2811LED::getState() {
  return theState;
}

/*****************************************************************
 * returns true if the led is on
 */
boolean   WS2811LED::isOn() {
  return theState == LEDState::ON;
}

/*****************************************************************
 * when destroyed, sets LED off and un-sets observer.
 */
WS2811LED::~WS2811LED() {
    setOff(false);
    setObserver(nullptr);
}

/*****************************************************************
 * nothing to do here, but has to be defined
 */
LEDObserver::~LEDObserver() {
    // more to do in descendants
}
```