```
/*
 * Defines the functionality for the WS2811LEDs class.
 *
 * Version 0.0   2022.11.25 LWH created
 *
 * Copyright (c) 2022, LWH brainware. All rights reserved.
 */
#define DEBUG         0

#include <LWHlog.h>
#include "FastLED.h"
#include "pixeltypes.h"
#include "WS2811LED.h"
#include "WS2811LEDChain.h"
#include <EEPROM.h>

/*******************************************************************
/*******************************************************************
 * Creates a new WS2811LEDs instance.
 * Does not add hardware control!
 * Controls the given number of LEDs (3 per controller)
 * keeps nominal brightnesses starting from aadr in EEPROM
 */
WS2811LEDChain::WS2811LEDChain(uint8_t achainidx, uint8_t anumleds, uint8_t aadr) {
  chainIdx    = achainidx;
  numLEDs     = anumleds;
  if (numLEDs > maxleds) {
    numLEDs   = maxleds;
    debugerr("ERROR: %i LEDs can't be handled by %i controllers.\n"
          "Only the first %i will be addressed!\n", anumleds, CONTROLLER_COUNT, maxleds);
  }
  adrEEPROM   = aadr;

  //  create default LED instances
  for (uint8_t i=0; i < numLEDs; i++) {
//     debugfine("adding %i of %i (%i)\n", i, numLEDs, adrEEPROM);
    addLEDObject(new WS2811LED(), i);
  }

  setTeaching(false);
  firstLED();
};

/*******************************************************************
 * adds the given LED object at the given index
 */
void   WS2811LEDChain::addLEDObject(WS2811LED* aled, uint8_t aidx) {
  if (aidx < numLEDs) {
    if (theLEDs[aidx] != nullptr) {
      theLEDs[aidx]->setObserver(nullptr);
    }
    theLEDs[aidx] = aled;
    theLEDs[aidx]->setIdx(chainIdx, aidx);
    aled->setObserver(this);
  }
};

/*******************************************************************
 * returns the given LEDs brightness from the CRGB struct.
 */
uint8_t WS2811LEDChain::getLEDCRGB(uint8_t aled)  {
  int   controller  = aled / 3;
  int   col         = aled % 3;
  switch (col) {
  case  0:
    return  controllers[controller].r;
    break;
  case  1:
    return  controllers[controller].g;
    break;
  case  2:
    return  controllers[controller].b;
    break;
  }
  return 0;
}
/*******************************************************************
 * transfers the given LED brightness into the CRGB struct.
 */
void  WS2811LEDChain::setLEDCRGB(uint8_t  aled, uint8_t  abrite)  {
  uint8_t    controller  = aled / 3;
  uint8_t    col         = aled % 3;
  debugfine("ch %i c %i led %i set to %i\n", chainIdx, controller, col, abrite);
  switch (col) {
  case  0:
    controllers[controller].r  = abrite;
    break;
  case  1:
    controllers[controller].g  = abrite;
    break;
  case  2:
    controllers[controller].b  = abrite;
    break;
  }
};

/*******************************************************************
 * updates all CRGBs from the LED instances.
 */
void  WS2811LEDChain::updateCRGB()  {
    for (uint8_t  i = 0; i < numLEDs; i++) {
      WS2811LED* curled   = theLEDs[i];
      setLEDCRGB(i, curled->getCurrentBrightness());
    }
};

/*******************************************************************
 * sets the current led to the first.
 * returns false if no leds.
 */
bool  WS2811LEDChain::firstLED()  {
  currentLEDIdx = 0;
  if (numLEDs > 0)  {
    debugfine("first  led  true  (%i)\n", currentLEDIdx);
    return true;
  }
  debugfinefix("first  led  false\n");
  return false;
}

/*******************************************************************
 * Returns the current led
 */
WS2811LED*  WS2811LEDChain::getCurrentLED() {
  return theLEDs[currentLEDIdx];
};

/*******************************************************************
 * Returns the given LED or null, if no such LED
 */
WS2811LED*  WS2811LEDChain::getLED(uint8_t  aled) {
  if (aled < numLEDs) {
    return theLEDs[aled];
  } else {
    return nullptr;
  }
};

/*******************************************************************
 * Selects the next current LED and returns true, if successful,
 * returns false, if no more LEDs
 */
```

```
bool  WS2811LEDChain::hasnext() {
  if (currentLEDIdx < numLEDs-1) {
    currentLEDIdx++;
    return true;
  }
  return false;
};

/*******************************************************************
 * lops all the leds and updates the LED controllers, if required
 */
void  WS2811LEDChain::loop()  {
  for (uint8_t  i=0; i < numLEDs; i++) {
    theLEDs[i]->loop();
  }
  if (updaterequired) {
    debugfinefix("update  required...\n");
    update();
  }
}
/*******************************************************************
 * Returns the current brightness (0..255) of the current LED .
 */
uint8_t  WS2811LEDChain::getLedBrightness()  {
  return  getLedBrightness(currentLEDIdx);
};

/*******************************************************************
 * Sets the brightness of the current LED to the given value (0..255)
 * Does not update the LEDs. Call update() for that.
 */
void  WS2811LEDChain::setNominalBrightness(uint8_t  abrite)  {
  setNominalBrightness(currentLEDIdx, abrite);
}

/*******************************************************************
 * Returns the current brightness (0..255) of the given LED .
 */
uint8_t  WS2811LEDChain::getLedBrightness(uint8_t  aled)  {
  return theLEDs[aled]->getCurrentBrightness();
};

/*******************************************************************
 * Sets the nominal brightness of the given LED to the given value (0..255)
 * Does not update the LEDs. Call update() for that.
 */
void    WS2811LEDChain::setNominalBrightness(uint8_t  aled, uint8_t  abrite)  {
  theLEDs[aled]->setNominalBrightness(abrite);
};

/*******************************************************************
 * Notifies the observer that a LED has changed
 */
void    WS2811LEDChain::updateFromLED(bool now) {
  debugfinefix("A LED has changed\n");
  updaterequired  = true;
  //some action
  if (now) {
    update();
  }
};

/*******************************************************************
 * Updates all LEDs
 */
void    WS2811LEDChain::update()  {
  for (uint8_t  i = 0; i < numLEDs; i++) {
    if (isTeaching) {
      setLEDCRGB(i, theLEDs[i]->getNominalBrightness());
    } else {
      setLEDCRGB(i, theLEDs[i]->getCurrentBrightness());
```

```cpp
    }
  }
  FastLED.show();
  updaterequired  = false;
};

/*****************************************************************
 * switches all LEDs off
 */
void    WS2811LEDChain::allOff(bool animate)  {
  for (uint8_t  i = 0; i < numLEDs; i++) {
    theLEDs[i]->setOff(animate);
  }
  update();
};

/*****************************************************************
 * sets teach mode
 */
void  WS2811LEDChain::setTeaching(bool ateach)  {
  isTeaching  = ateach;
}
/*****************************************************************
 * Blinks the current LED acount times.
 */
void    WS2811LEDChain::blinkLed(uint8_t  acount)  {
  blinkLed(currentLEDIdx, acount);
};

/*****************************************************************
 * Blinks the given LED acount times.
 * Hardware only, does not use LED objects.
 */
void    WS2811LEDChain::blinkLed(uint8_t  aled, uint8_t  acount)  {
  uint8_t    lastbrite  = getLEDCRGB(aled);
  debug("blinking %i/%i\n", chainIdx, aled);
  for (uint8_t  i = 0; i < acount; i++) {
    setLEDCRGB(aled, 255);
    FastLED.show();
    delay(100);
    setLEDCRGB(aled, 0);
    FastLED.show();
    delay(200);
  }
  setLEDCRGB(aled, lastbrite);
  FastLED.show();
};

/*****************************************************************
 * Loads all brightness values from the eeprom
 * when all are loaded, updates LEDs
 */
void  WS2811LEDChain::load()  {
  debug("loading from %i ff\n", adrEEPROM);
  uint8_t  tmpbr   = 0;
  for (uint8_t  i = 0; i < numLEDs; i++) {
    tmpbr   = EEPROM.read(adrEEPROM + i);
    if (tmpbr == 255) {
      // if never stored (EEPROM default = 255), set an average
      tmpbr= 128;
    }
    setNominalBrightness(i, tmpbr);
  }
  update();
};

/*****************************************************************
 * Stores all brightness values to the eeprom
 */
void  WS2811LEDChain::store() {
  debug("saving to %i ff\n", adrEEPROM);
  for (uint8_t  i = 0; i < numLEDs; i++) {
    EEPROM.write(adrEEPROM + i, theLEDs[i]->getNominalBrightness());
  }
};

/*******************************************************************
 * switches all LEDs off and destroys them
 */
WS2811LEDChain::~WS2811LEDChain()   {
  for (uint8_t  i = 0; i < numLEDs; i++) {
    theLEDs[i]->setOff(false);
  }
  update();
  delete[] theLEDs;
}
```