```
/*
 * Defines the functionality for the Drehknopf class.
 *
 * Version 0.0  2022.02.13 LWH created
 *
 * Copyright (c) 2022, LWH brainware. All rights reserved.
 */
#define DEBUG         0

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

/*****************************************************************
 * Creates a new Drehknopf instance.
 * Sets the analog input to the given pin.
 * Sets the state to idle.
 */
Drehknopf::Drehknopf(int apin) {
  analogpin   = apin;         // the pin needs not be set as analog input
  stablecnt   = 0;
  hysteresis  = 20;
  nexttime    = millis();
  // default range values
  setValRange(0, 1023);
  setHomeValRange(480, 540);
  setPosRange(0, 100);
  pinMode(analogpin, INPUT);  // just for clarity
  finished();                  // start idle
  debug("Drehknopf set up for pin %i\n", apin);
}

/*****************************************************************
 * returns the current analog value
 */
int Drehknopf::readValue(void) {
  return analogRead(analogpin);
}

/*****************************************************************
 * reads the analog input, checks against last value and acts accordingly
 */
void Drehknopf::loop(void)  {
  int           current   = readValue();
  // do the evaluation only every 50 ms
  if (millis() < nexttime) {
    return;
  }
  nexttime  = millis() + 50;
  switch (turnstate) {
    case turningState::IDLE:
    case turningState::HOMESELECTED:
      // if the dial has moved considerably
      if ((abs(current - lastvalue) > hysteresis))  {
        turnstate = turningState::CHANGING;
        debug("potentiometer started turning from %i\n", current);
        lastvalue = current;
        stablecnt = 0;
      }
      break;
    case turningState::CHANGING:
//      debug("current value: %i(%i)\n", current, lastvalue);
      // if the dial does not move any more
      if ((abs(current - lastvalue) < 5))  {
        stablecnt++;
        if (stablecnt > 5)  {
          turnstate = turningState::SELECTED;
          debug("potentiometer stopped turning at %i\n", current);
        }
      } else  {
        stablecnt = 0;
        lastvalue = current;
```

```
      }
      break;
    case turningState::SELECTED:
      if ((lastvalue > homemin) && (lastvalue < homemax)) {
          turnstate = turningState::HOMESELECTED;
      }
      break;
    default:
      debug("ERROR: no such state!");
  }
}

/*****************************************************************
 * returns the current turning state.
 */
turningState Drehknopf::getTurnState(void) {
  return  turnstate;
}

/*****************************************************************
 * returns true if the current turning state is "selected"
 */
bool    Drehknopf::isSelected(void) {
  return (turnstate == turningState::SELECTED);
}

/*****************************************************************
 * sets the state back to idle after processing
 * and memorizes the current position
 */
void    Drehknopf::finished(void) {
  lastvalue = readValue();
  turnstate = turningState::IDLE;
}

/*****************************************************************
 * returns the last stable value
 */
int     Drehknopf::getValue(void) {
  return  lastvalue;
}

/*****************************************************************
 * sets the value range expected for leftmost position (avmin) and rightmost position (avmax)
 * defaults to 0..1023
 */
void    Drehknopf::setValRange(int avmin, int avmax) {
  valmin  = avmin;
  valmax  = avmax;
}

/*****************************************************************
 * sets the value range within which the home position is selected
 * defaults to 480..540
 */
void    Drehknopf::setHomeValRange(int ahvmin, int ahvmax) {
  homemin  = ahvmin;
  homemax  = ahvmax;
}

/*****************************************************************
 * sets the value range expected for leftmost position (avmin) and rightmost position (avmax)
 * defaults to 0..100
 */
void    Drehknopf::setPosRange(long apmin, long apmax) {
  posmin  = apmin;
  posmax  = apmax;
}

/*****************************************************************
 * returns the position within the defined range represented by the last stable value
```

```
 */
long     Drehknopf::getPosition(void) {
  long result = map(lastvalue, valmin, valmax, posmin, posmax);
//  float v = (float) (lastvalue - valmin) / (float) (valmax - valmin);
//  long  result  = v * (posmax - posmin) + posmin;
  debug("Potentiometer value %i used as position %li\n", lastvalue, result);
  return result;
}
```