```
#include "weight_sensor.h"
#include <ArduinoNvs.h>
#include <EEPROM.h>

#define MAX_SAMPLES 20
const int calVal_eepromAdress = 0;
unsigned long t = 0;

bool WeightSensor::setNvsCalibrationVal(float calibVal) {
 bool ok = true;
 ok &= NVS.setFloat("calibVal", calibVal); // Stores the integer value 23 into the key named "myInt" on the NVS

 String data1 = "AA55";
 ok &= NVS.setString("validity", data1); // Store the data value into the key named "myString" on the NVS

 return ok;
}

bool WeightSensor::getNvsCalibrationVal(float* calibVal) {
 // bool ok;
 // ok = NVS.setInt ("myInt", 23);
 // Serial.print("is ok: ");
 // Serial.println(ok);
 *calibVal = NVS.getFloat("calibVal");
 Serial.print("read calibValue value: ");
 Serial.println(*calibVal);
 String validity = NVS.getString("validity");
 Serial.print("validity: ");
 Serial.println(validity);

 if ( validity == "AA55" ) {
   return true;
 }

 return false;
}

void WeightSensor::refresh()
{
 static boolean newDataReady = 0;
```

```
 const int serialPrintInterval = 0; //increase value to slow down serial print
activity

 // check for new data/start next conversion:
 if (this->update()) newDataReady = true;

 // get smoothed value from the dataset:
 if (newDataReady) {
   if (millis() > t + serialPrintInterval) {
     float i = this->getData();
     //Serial.print("Load_cell output val: ");
     //Serial.println(i);
     newDataReady = 0;
     t = millis();
   }
 // Serial.println("End change calibration value");
 //Serial.println("***");
 this->update(); // retrieves data from the load cell
 float weight = this->getData(); // get output value
 this->setWeight(weight);
 weight = calculate_avg_weight(weight);
 this->setAverageWeight(weight);
 char weight_str[16];
 dtostrf(weight, 4, 2, weight_str); //convert float to string with 2 decimals after
point.

//Serial.print("one reading:\t");
 //Serial.print(scale.get_units(), 1);
 //Serial.print("\t| average:\t");
 //Serial.println(scale.get_units(10), 5);

 //delay(1000);
 }

 // receive command from serial terminal
 if (Serial.available() > 0) {
   char inByte = Serial.read();
   if (inByte == 't') this->tareNoDelay(); //tare
   else if (inByte == 'r') this->calibrate(); //calibrate
   else if (inByte == 'c') this->changeSavedCalFactor(); //edit calibration value
manually
 }
```

```
 // check if last tare operation is complete
 if (this->getTareStatus() == true) {
   Serial.println("Tare complete");
 }
}

float WeightSensor::calculate_avg_weight(float weight)
{
   static float prev_w = 0;
   static float avg_w = 0;
   static int valid_samples = 0;
   static unsigned long last_valid_sample_time = 0;

   if (fabs( (float)(prev_w - weight)) > (0.05 * avg_w + 0.00001)) //
   {
       if ((last_valid_sample_time == 0) ||
           ((millis() - last_valid_sample_time) >= 5000))
       {
           /* Last valid sample is recieved 5 seconds back.
              So, ignore the previous values and start averaging again. */
           avg_w = weight;
           valid_samples = 1;
       }
       else
       {
           return avg_w;
       }
   }
   else
   {
       valid_samples++;
       if (valid_samples <= MAX_SAMPLES)
       {
           avg_w += weight;
           avg_w *= 0.5;
       }
       else
       {
           /* Samples are satbilized now. No need to avg new values again */
       }
```

```
        last_valid_sample_time = millis();
    }

    prev_w = weight;
    return avg_w;
}

void WeightSensor::initialize()
{
 this->begin();
 //this->setReverseOutput(); //uncomment to turn a negative output value to positive
 unsigned long stabilizingtime = 2000; // preciscion right after power-up can be improved by adding a few seconds of stabilizing time
 boolean _tare = true; //set this to false if you don't want tare to be performed in the next step
 this->start(stabilizingtime, _tare);
 if (this->getTareTimeoutFlag() || this->getSignalTimeoutFlag()) {
   Serial.println("Timeout, check MCU>HX711 wiring and pin designations");
   while (1);
 }
 else {
   this->setCalibrationFactor(1.0); // user set calibration value (float), initial value 1.0 may be used for this sketch
   Serial.println("Startup is complete");
 }
 while (!this->update());

 bool calibration_required = true;

 NVS.begin();
 float prevCalibVal;
 if ( getNvsCalibrationVal(&prevCalibVal) ) {
   calibration_required = false;
   this->setCalibrationFactor(prevCalibVal);
 }

 if (calibration_required)
 {
   calibrate(); //start calibration procedure
 }
}
```

```
void WeightSensor::calibrate() {
 Serial.println("***");
 Serial.println("Start calibration:");
 Serial.println("Place the load cell an a level stable surface.");
 Serial.println("Remove any load applied to the load cell.");
 Serial.println("Send 't' from serial monitor to set the tare offset.");

 boolean _resume = false;
 while (_resume == false) {
   this->update();
   if (Serial.available() > 0) {
     if (Serial.available() > 0) {
       char inByte = Serial.read();
       if (inByte == 't') this->tareNoDelay();
     }
   }
   if (this->getTareStatus() == true) {
     Serial.println("Tare complete");
     _resume = true;
   }
 }

 Serial.println("Now, place your known mass on the loadcell.");
 Serial.println("Then send the weight of this mass (i.e. 100.0) from serial
monitor.");

 float known_mass = 0;
 _resume = false;
 while (_resume == false) {
   this->update();
   if (Serial.available() > 0) {
     known_mass = Serial.parseFloat();
     if (known_mass != 0) {
       Serial.print("Known mass is: ");
       Serial.println(known_mass);
       _resume = true;
     }
   }
 }

 this->refreshDataSet(); //refresh the dataset to be sure that the known mass is
measured correct
```

```
 float newCalibrationValue = this->getNewCalibration(known_mass); //get the new
calibration value

 setCalibrationFactor(newCalibrationValue);

 if ( !setNvsCalibrationVal(newCalibrationValue) ) {
   Serial.println("Saving to NVS memory failed");
 }

 Serial.print("New calibration value has been set to: ");
 Serial.print(newCalibrationValue);
 Serial.println(", use this as calibration value (calFactor) in your project
sketch.");
 Serial.print("Save this value to EEPROM adress ");
 Serial.print(calVal_eepromAdress);
 Serial.println("? y/n");

 _resume = false;
 while (_resume == false) {
   if (Serial.available() > 0) {
     char inByte = Serial.read();
     if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
       EEPROM.begin(512);
#endif
       EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
       EEPROM.commit();
#endif
       EEPROM.get(calVal_eepromAdress, newCalibrationValue);
       Serial.print("Value ");
       Serial.print(newCalibrationValue);
       Serial.print(" saved to EEPROM address: ");
       Serial.println(calVal_eepromAdress);
       _resume = true;

     }
     else if (inByte == 'n') {
       Serial.println("Value not saved to EEPROM");
       _resume = true;
     }
   }
```

```
 }

 Serial.println("End calibration");
 Serial.println("***");
 Serial.println("To re-calibrate, send 'r' from serial monitor.");
 Serial.println("For manual edit of the calibration value, send 'c' from serial
monitor.");
 Serial.println("***");
}

void WeightSensor::changeSavedCalFactor() {
 float oldCalibrationValue = this->getCalFactor();
 boolean _resume = false;
 Serial.println("***");
 Serial.print("Current value is: ");
 Serial.println(oldCalibrationValue);
 Serial.println("Now, send the new value from serial monitor, i.e. 696.0");
 float newCalibrationValue;
 while (_resume == false) {
   if (Serial.available() > 0) {
     newCalibrationValue = Serial.parseFloat();
     if (newCalibrationValue != 0) {
       Serial.print("New calibration value is: ");
       Serial.println(newCalibrationValue);
       this->setCalibrationFactor(newCalibrationValue);
       _resume = true;
     }
   }
 }
 _resume = false;
 Serial.print("Save this value to EEPROM adress ");
 Serial.print(calVal_eepromAdress);
 Serial.println("? y/n");
 while (_resume == false) {
   if (Serial.available() > 0) {
     char inByte = Serial.read();
     if (inByte == 'y') {
#if defined(ESP8266)|| defined(ESP32)
       EEPROM.begin(512);
#endif
       EEPROM.put(calVal_eepromAdress, newCalibrationValue);
#if defined(ESP8266)|| defined(ESP32)
```

```
        EEPROM.commit();
#endif
        EEPROM.get(calVal_eepromAdress, newCalibrationValue);
        Serial.print("Value ");
        Serial.print(newCalibrationValue);
        Serial.print(" saved to EEPROM address: ");
        Serial.println(calVal_eepromAdress);
        _resume = true;
      }
      else if (inByte == 'n') {
        Serial.println("Value not saved to EEPROM");
        _resume = true;
      }
    }
  }

}
```