1.1 Implementation

Here is how the code implements this temperature compensation.

Based on what we have explained before, we need to keep memory of all past external temperature data that can have an impact on the load cell temperature, given its thermal inertia. Knowing we take a readout of temperature at each measurement, we need to decide on how many past measurements we keep memory of the temperature. Arbitrary and knowing on one hand that our load cell mounted to its frame has a time constant of 17mn and on the other hand that most of the time we will choose to take a measure every 15mn, we have decided to keep the last 5 measures. With our parameters, the temperature of 5 measures ago will have 99% finished to influence the current temperature of the load cell. Longer interval between measures won't make any problem but shorter might.

Three arrays are declared in the code for the calculations:

```
double deltaText[5] = {0,0,0,0,0}; // an array containing the last 5 external temperature changes
double deltaTlc[5] = {0,0,0,0,0};  // an array containing the last 5 load cell tempperature changes
double Text[6] = {T_EXT_INIT,T_EXT_INIT,T_EXT_INIT,T_EXT_INIT,T_EXT_INIT,T_EXT_INIT}; // an array containing the last 6 external temperatures
```

Each time a new temperature measure is taken, two arrays (`Text` and `deltaText`) are right shifted (the last value is dropped, and the first one is updated according to this last measure):

- `Text[0]` is loaded with the last external temperature measure,
- `deltaText[0]` is loaded with the difference (current temperature minus temperature at previous measure)

`deltaTlc` is then calcultated based on external temperature changes and time elapsed since those temperature changes:

- `deltaT_LC[0]` is loaded with the load cell temperature change resulting from a temperature step equal current temperature minus previous measurement temperature:

  $\Delta T_{LC} = \Delta T_{ext}\left(1 - e^{\frac{-\sigma}{C}t}\right)$ where *t* is the time difference between current measure and previous one and $\frac{\sigma}{C}$ is the inverse of the time constant (`TIMECONSTANT`) as specified in file `scale-parameters.h`
- `deltaT_LC[1]` is loaded with the same calculation replacing `t` by `2t` and taking into account the previous temperature change
- And so on and so forth until is `deltaT_LC[4]` calculated

Based on the calculations that we have explained earlier, the actual load cell temperature is the external temperature 5*t minutes ago (t being the time difference between two consecutive measures) plus the sum of each element of this array deltaT_LC.

scale-parameters.h file contains a few additional important parameters: WEIGHTSLOPE is one and is the value (in grams) by which the weight measurement varies when the load cell temperature increases by 1 degree (should be measured as part of the calibration process by taking note of weight measurement difference – with the same mass applied to the scale – at two different temperatures, differing from each other as much as possible, for precision sake).

As a result of the calculations we have just made, we now have the actual load cell temperature; if we know the initial load cell temperature at which the scale calibration was done, we can calculate the difference between those two temperatures, multiply this difference by the parameter WEIGHTSLOPE and this value (current_temp – calib_temp)*WEIGHTSLOPE

would be the number of grams that we need to substract from the last weight measurement to get the actual (temperature compensated) weight.

The load cell temperature at which the calibration was done is also a constant that is stored in file scale-parameters.h. Its name is CALIBTEMP. In real life, if this temperature is not known precisely, it can be approximated because in fact we don’t care too much about the precision of this value because we are more interested in weight variation than in absolute weight (in other terms it’s not a big deal if we make an error of a few grams, as long as this error doesn’t vary over time and doesn’t hinder the compensation calculation).