University of Otago, 2021 Semester 2 (New Zealand)
COSC326 (Effective Programming)
Etude-9: Pulses Counting

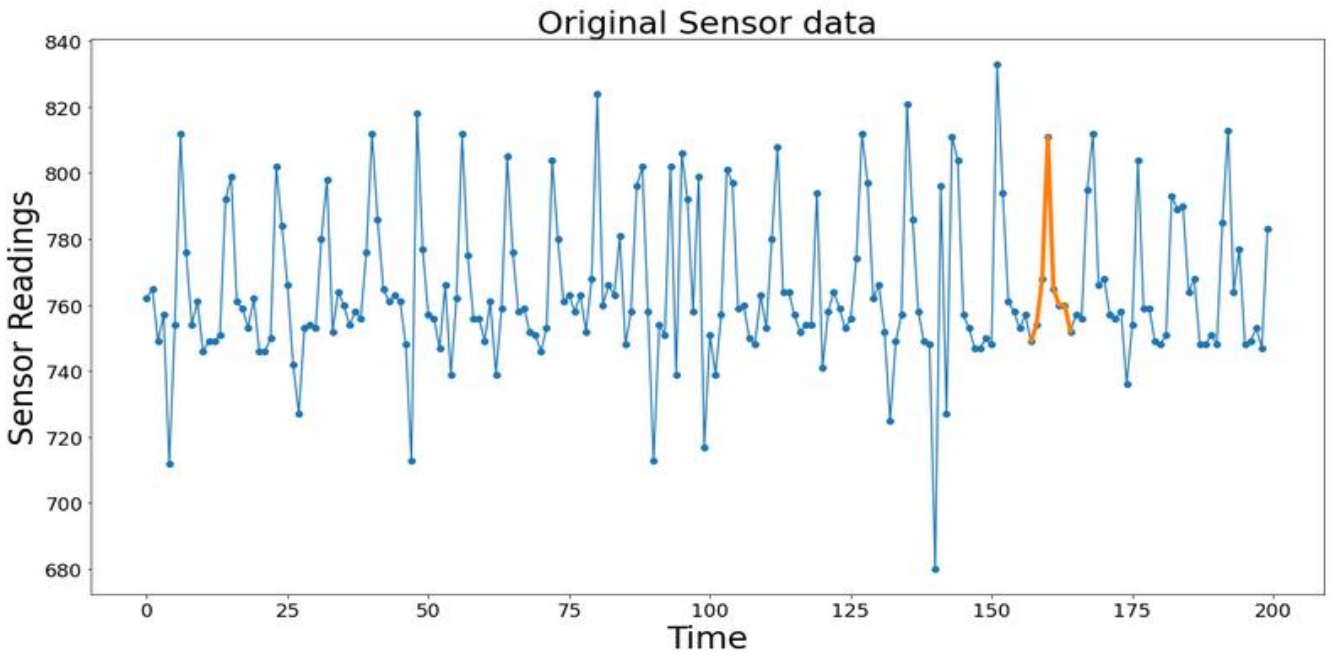
Student Names: Valentin Kiselev, Leo Venn

COSC 326
Etude 9
("Pulses Counting")

REPORT

In many Internet of Things (IoT), there are various sensors collecting data like pulses and motion. For this exercise we were given samples of pulses directly collected from a pulse's sensor. The data was collected with a sampling rate of 10Hz. The data is given as a dataset text file which contains a list of recorded pulse data, i.e. samples of pulses directly collected from a pulse's sensor. The data in the file is represented as integers in a single column. The main task of this project was to write a program which will read in the pulse data from a given dataset file of sensor readings and accurately count the number of pulses without considering the noisy interference which was mixed with actual sensor readings.

To filter out the noise from the signal, the data from an example dataset was first visualized as seen in example bellow:



The above graph shows plotting of the 200 sensor readings from one of the dataset files. After plotting this dataset, an attempt was made to visually identify a common generalized pattern of a pulse wave which had minimum amount of noise. Such patten was chosen between 155 and 165 time interval as shown in the orange color on the plot above. After identifying this pattern, we will record its sensor readings (in this case 8 readings) to store them for future processing. However, as seen on the plot above, the chosen wave patten is still not perfect, especially in terms of symmetry. Hence, now our chosen pattern will need to be improved to match our needs. Since we don't care about the exact shape of the pulses as we mainly interested in the peaks of the waves, we need to improve our pattern so it will look like the peaks of the pulses we are interested in. Therefore, the recorded values of our chosen pattern were manually modified and polished to make its shape look like a proper pulse shape we are interested in and hence to improve its filter capabilities.

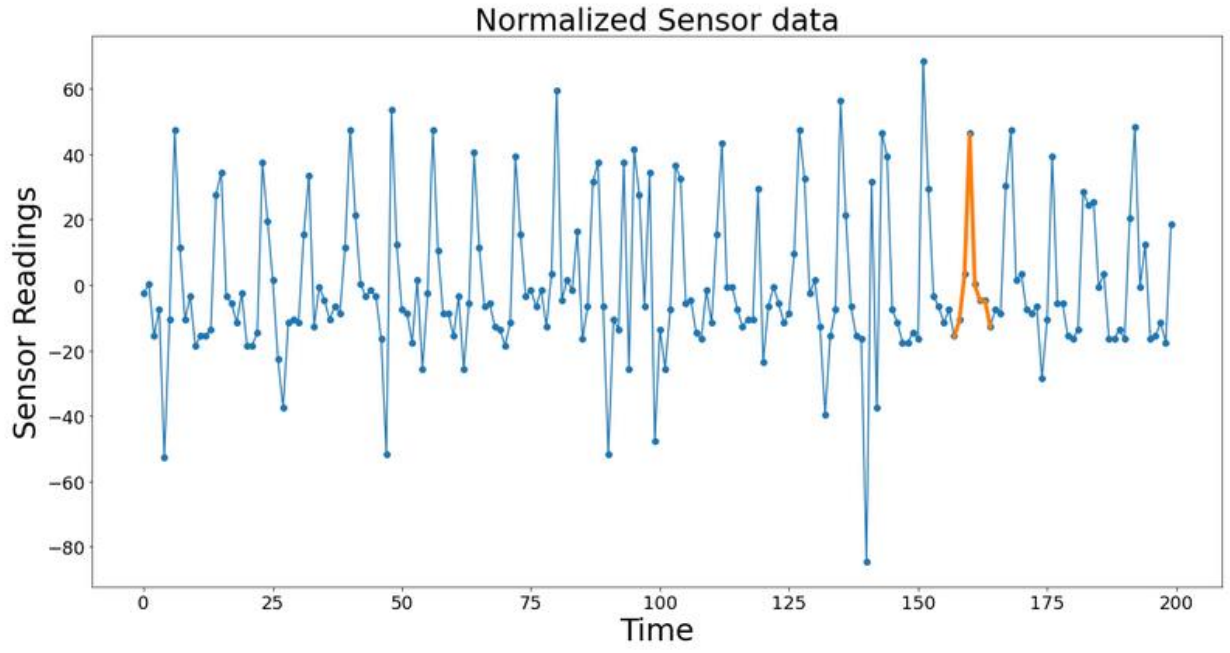
Then all sensor's reading data was normalized (incl. identified pattern readings) so its values would spin around zero y value. Data normalization was done by adding up all readings and then dividing them by the total number of readings. Then the resulted mean value was subtracted from each original dataset point. Now the resulted new list of values represented the normalized sensor readings. Hence normalization procedure can be summarized as follows:

(i). Find mean using this formula: $Mean = \frac{\sum_{i=1}^{N} yi}{N}$

Where y_i are the raw values recorded from the sensor, and N in the total number of recordings. The mean value needs to be found for both, the whole original dataset readings and the readings of our identified general pulse pattern. Hence in this example, N will be 200 for the whole dataset and for general pattern N will be 8.

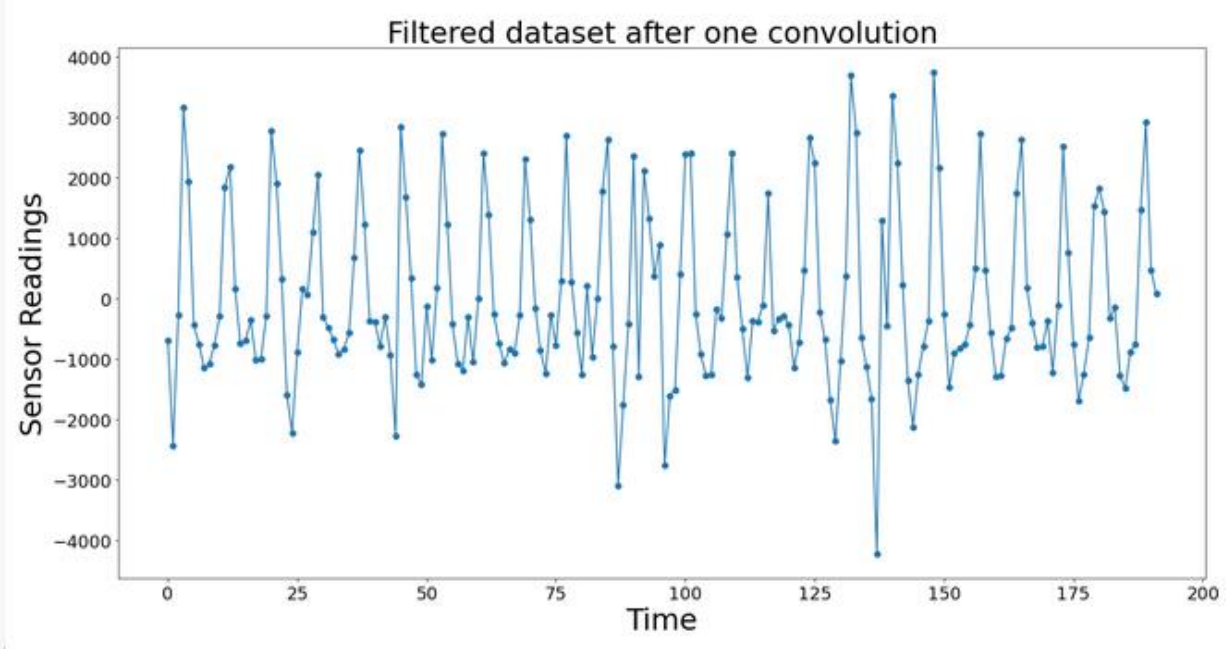
(ii). To finally normalize the data, subtract "mean value" of original dataset from each value in the original dataset. Then repeat the same procedure with general pulse pattern readings, i.e. subtract "mean value" of general pulse pattern readings from each value in the general pulse pattern.

The bellow plot shows the normalized dataset sensor readings values (including the normalized readings values of our chosen general pulse pattern as shown by orange color):

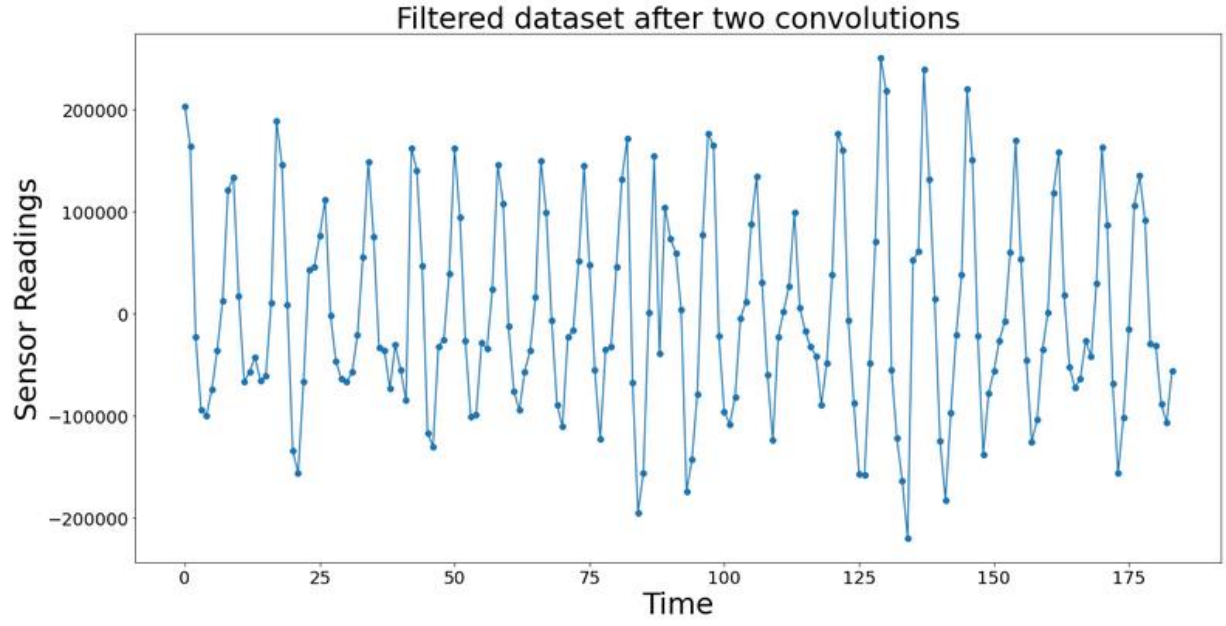


The normalized values of our found general pulse pattern are now can be used as the convolution filter (aka kernel) to filter out the noise and enhanced any pulse waves which are more similar to our general pulse pattern. In this example the convolution was performed by moving 8 value sliding window across the normalized dataset of all recorded sensor readings and during each move computing a dot product between these sliding window values and our

found convolution filter. By performing such convolution, it will result in amplifying any waves which had more similarities with our general pulse pattern we have found and also reducing any waves which were less similar to this general patter. The bellow plot shows the resulting filtered dataset after performing one convolution over it:



To improve filtering results the convolution was performed again for the second time. Hence it will amplify the “true” pulse signals even more and reduce noisy data even further. The bellow plot shows second convolution run on the previously convoluted dataset:



As can be seen on the plot above, the repeated convolution procedure has significantly enhanced the proper pulse signals and also noticeably has reduced the noisy data by suppressing any pulses which considerably deviated from the general pulse pattern that we have defined for our kernel filter earlier. Finally, to count pulses we just need to count number of times any given pulse crosses a median line (and in our case the median line will be around y value of 0). Because each pulse crosses a median line twice by going up and down, we need to divide this total count by half which eventually will give us the total number of actual pulses recorded by the sensos. Hence by using our custom convolution filter, which we have deducted from visually analyzing the original dataset, we can reduce interference from noise, enhance original proper pulse signals and therefore accurately find the total number of pulses which were actually recorded by the sensor.