HW4: Real-time smartphone sensor data processing. Accelerometer/Gyroscope applications

The purpose of this assignment is to help you get experience with Android application programming using hardware sensors in the phone. You will also deal with the processing of the sensory data in real time.

Plese note that, while you are going to build only two functionalities in this assignment, they could apply to a wider range of applications. For example, for physical therapy or safety monitoring applications, serious games, and simply recreational activities.

Description

This assignment should take the form of an Android App, implementing the two following functionalities

1. **Shake detection:**
   The App should be able to detect when the user shakes the phone, and it should start the **shaking-triggered counter** (described below) when the shake is detected.
2. **Step (movement)-triggered counter:**
   When the phone shaking is detected, the counter should start, but the counter can operated in two modes:
   a) **User defined mode:** the user should define the maximum value, ***nmax***. When the counter reaches *nmax*, play a sound of your choice to notify the user. To play the sound, you can use the *MediaPlayer* library, or just play a tone using an instance of the class *ToneGenerator*.
   There should be a stop button to stop the counter if the user does not finish the *nmax* repetitions. To start again, shake the phone.
   b) **Free mode**: the user just performs the shaking movement, and the app starts counting movement/shaking, and displays the count. There should be a stop button to stop the counter. To start again, shake the phone.

Additional considerations:

- We will assume that the phone is in a portrait mode all the time. So, it is accepted to lose the state of the activity upon rotation (we will not rotate the phone to check the performance).
- You should follow the good practice for sensors studied in Lecture 5. Remember to release the sensor when the App is not active and running.
- Assume that the user will not be doing the movements very quickly. At least one or more second per shake.

Submission

Submit your assignment using Github and Canvas, under the Assignment HW4. The due date is shown in Canvas. You will want to push the following to your personal Github repo:

- Android Studio Project folder
- README.txt or README.md

The README file should contain your name and email address, along with the name of your app and a brief description of it. Specifically, we are interested in the description of the photo editing that your app is performing. Finally, please include any special instructions that the user might need to know in order to use your app properly (if there are any). For example:

Tamara Bonaci <tbonaci@uw.edu>
Sorting Hat - This app displays the sorting hat and a picture of the user. It returns the house where the user belongs.

Grading (0-100 points)

Your submission will be graded by building and running it and evaluating its functionality. Your code will not be graded on style, but we still encourage you to follow good overall coding style for your own good.

- 0-20 points: the app compiles, builds and loads without errors.
- 0-30 points: Shake detector works correctly.
- 0-45 points: Pull-ups detector detects movements with "certain" accuracy. We will test the detector by simulating movements with the phone in hands without rotating the phone. The 2 operation modes are implemented.
- 0-5 points: The User Interface is simple and well organized.
- Advice: please do not spend too much time on a perfect UI, but make it functional. Widgets should have some meaningful function; all widgets appear on the screen without being cropped. The app is intuitive for the user to use.

Getting help and resources

Pull-up counter detection based on simple step counting

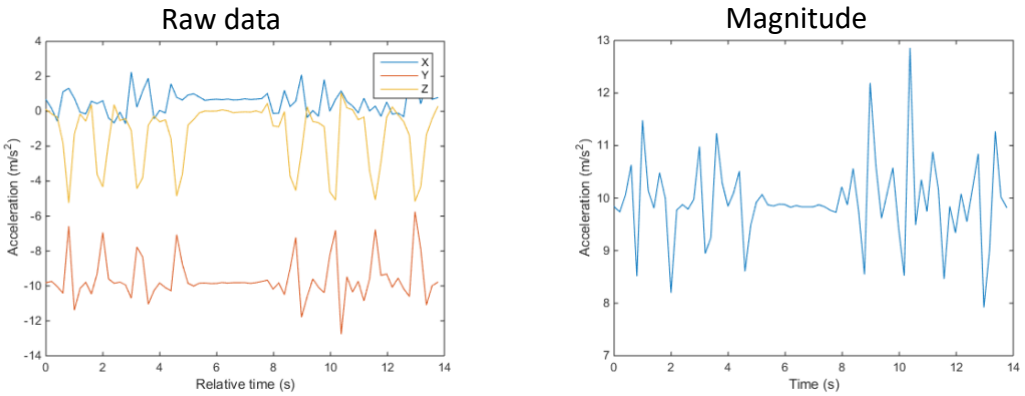
There are three basic methods to count the number of steps based on the accelerometer data: peaks-finding, zero-crossing, and frequency domain. This document will cover the peak finding methodology, but you can implement any of the other methods if you like. Use this document as a guide, that provides a **simple first-approach method of step counting**. Remember that you are going to work with real time continuous samples read from the accelerometer.

1. Calculate the magnitude of the acceleration vector

To convert the XYZ acceleration vectors at each point in time into scalar values, the magnitude is calculated. This allows large changes in overall acceleration, such as steps taken while walking, to be detected regardless of device orientation.

$$r_{magnitude} = \sqrt{x_{accel}^2 + y_{accel}^2 + z_{accel}^2}$$

The magnitude of the accelerometer data when you, will look something like this:



- You can plot the accelerometer data using the real time GraphView to visualize the data in real time (refer to the App SensorGraphs in Canvas).
-

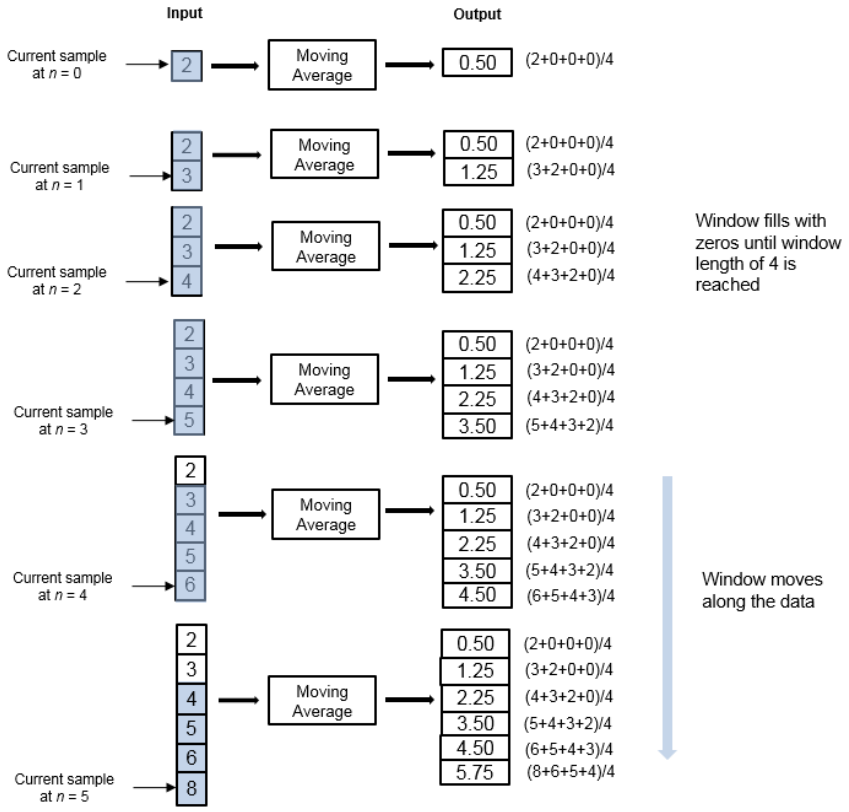
2. Subtract the mean

The acceleration magnitude is not zero-mean. Subtracting the mean from the data will remove any constant effects, such as gravity. One example of implementing that is to have an initial calibration function where you hold your board still for some time, calculate the average. And then use this average for your real-time processing.

3. Define a sliding window

As your signal gets longer and longer, it does not make sense to analyze all that data at once: it is infeasible for memory, and older information is usually less useful or unneeded. You will only want to look at the most recent chunk of data. When the next value comes, add the value to your list and get rid of the oldest one

4. Filtering: a simple smoothing filter is the moving average. The figure below provides a good illustration of this algorithm.



5. Counted number of peaks: the data will show peaks in acceleration magnitude. Each peak corresponds to a step being taken while walking. You should specify a threshold for the peak detection. This threshold should be tuned experimentally to match a person's level of movement while walking, hardness of floor surfaces, etc.

Possible implementation of a Sliding window of size 5

```
internal var slidingWindow = DoubleArray(5)

fun addToWindow(x: Double) {
    // Shift everything one to the left
    for (i in 1 until slidingWindow.size) {
        slidingWindow[i - 1] = slidingWindow[i]
    }

    // Add the new data point
    slidingWindow[slidingWindow.size - 1] = x

}
```