CS4222: Project Report (Group 9)

Jabir Shabbir Karachiwala A0123480N
Prateek Kumar A0123398Y
20-April-2015

*Introduction and Overview of the problem: **FitnessApp:***

Our Project idea is an Android app intended to be used in the Gymnasium. The intention is to use the sensors in the phone which is kept the (lower) pocket to measure acceleration for step counting. The user is able to invite friends/neighbors in the gym to a virtual "race" or on a treadmill. This communication happens over the DTN Forwarding Layer so it will work in multi-hop mode as well, although for our purposes single hop is sufficient in a regular-sized Gym with a single access point.

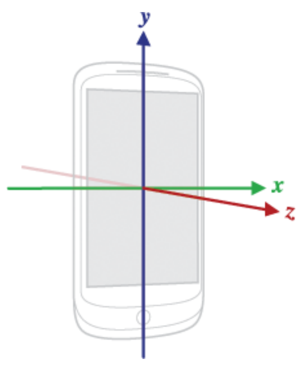
So any user opens the app and broadcasts this invite and some users respond by accepting it after which each phone begins the game. The Accelerometer is used for detecting when the user has taken a step and counts the number of steps. The app can also be used to calibrate by first asking the user to walk a known distance (say 100m which they enter in the app via text field) and then dividing that by the steps counted to get an average "distance per step". This is multiplied with step count when the game is running to get the current distance traversed by each player.

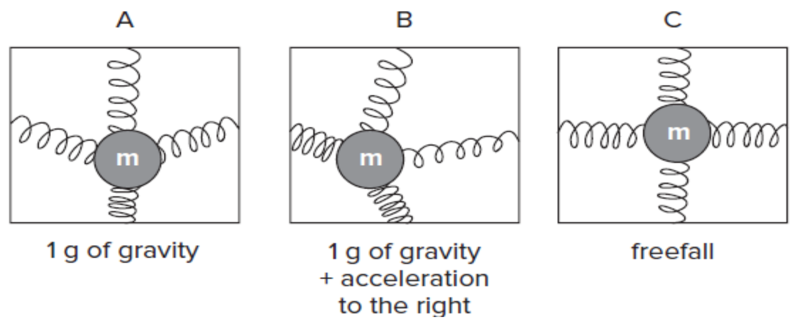
The scores of each user (step count/distance) in the game are repeatedly broadcast over the DTN Forwarding layer to each other. So each user periodically gets to know the distance everyone else has power-walked. The game ends after a certain amount of time (10 min) and the user at the top of the leader board is declared the winner who has walked the largest distance (thus exercised the most).

Features of Proposed System:

Step Counter:

This component uses the accelerometer sensor to figure out when the user has taken a step and counts it. The Accelerometer is a raw sensor as it provides the raw values of the acceleration sampled at a very fast rate, and along 3 axes – X, Y, and Z relative to the device co-ordinate system. The value along the Z-axes when the phone is stationary always has the absolute value of g due to gravity.





Networking:

The DTN forwarding layer was used for networking. The phones need to be able to send messages to each other and the app broadcasts over this Delay Tolerant Network periodically. The initial invite is broadcast which contains the leaders IMEI number for identification. The phones where the app is open accepts the invite and respond with their own IMEI identification numbers which are used to index into the scoreboards once the game starts. DTN is useful as messages are not frequent nor in real time.

Implementation :("Project" App in zip)

For the step counter a few techniques were first tested/coded – these are summarized below. It's worth noting that ALL algorithms, smoothing average filters and Kalman estimators are standard techniques and we've referenced the sources from where the equations were taken in the paragraphs as below.

Linear Double Integration:

Whenever the sensor event was delivered with the current acceleration value, its total value a=sqrt(x^2+y^2 z^2) was used with the linear equations of motion to calculate distance in that fraction of time. These distances were summed up and since they were automatically sampled, the running value accumulated was the distance travelled so far. The sampling rate was high enough that even if the phone was moved around and not in a straight line, the quantization was directly approximated in timeslice=dt. Since v=Integral(a.dt) and s=Integral(v.dt), the double integral was calculated by the app. Linear acceleration became a quadratic velocity and eventually a cubic distance.

The problem with this approach is/was that the error is also integrated as the sensor is very noisy. When practically implemented, it was almost impossible to discern the changes in distance (which kept increasing) due to walking as compared to when the phone was stationary – and this method was unworkable. We have included the zipped project if needed, although its code was not used in the final version.

Threshold Algorithm:

There are standard "threshold" algorithms for doing step detection that give a reasonably accurate step count, and one such method was implemented for this project. While the first step is to smooth out the data, the algorithm itself is roughly as follows. The phone is kept in the trouser/lower pocket so as to oscillate with the leg while walking naturally. Its orientation does not matter as some component of this oscillation will definitely fall along the Z-axis coming out of the phone screen, but we still recommend keeping it such that the screen faces directly forwards, or backwards (if say in the back pocket).

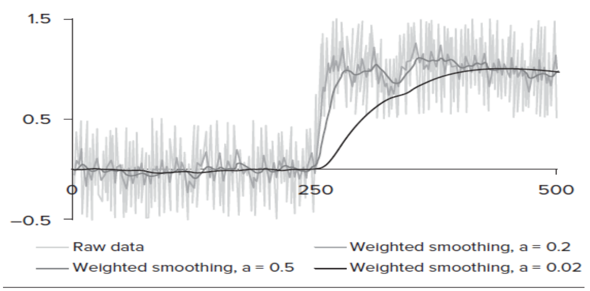
For each consecutive samplings of acceleration along the Z-axes, the absolute difference after smoothing is compared with a certain threshold. If the difference is above it, then we are at the end of a step and *the leg/thigh/knee is at the highest crest in the regular walking motion, either in the forward or backward direction. A step counter is incremented if so, indicating that a step has been identified.*

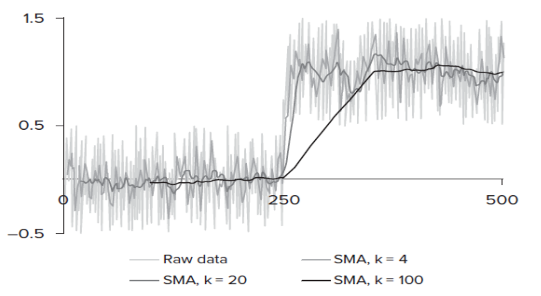
If it's below the threshold we ignore it since the sampling happens much faster than the duration of one step, although if the threshold isn't breached after a certain amount of time, we know that the user is not walking, and the status is changed from "walking" to "idling". The algorithm is similar to the one at this StackOverflow thread http://stackoverflow.com/questions/4993993/how-to-detect-walking-with-android-accelerometer and a related paper is http://www.enggjournals.com/ijcse/doc/IJCSE12-04-05-266.pdf.

Low Pass Smoothing Average Filter:

The accelerometer readings are very noisy as it's a raw sensor. When the phone is lying stationary on its back with the screen facing up, the values expected along the X,Y and Z axes should be {0,0,-g}. In reality, the values keep oscillating with ranges like -0.15 to 0.15 for X and Y and g+-0.15 along Z. Hence it's impossible to accurately measure a true change in acceleration just from this data without smoothing this out using a filter. A low pass filter was implemented which does average-smoothing based on a parameter weight "a" = 0.1f with results like below (representative image only):

float lowPass(float current, float last) return last * (1.0f - a) + current * a**;**





As we can see, the data is now relatively smooth enough for consecutive samples to have a difference in readings due to acceleration and not just noise. This filter was used in our code Project app.

Kalman Filter:

A Kalman Filter is actually an estimator which converges to the true value over time depending on whether the sensor variance is trusted more/less than the guess variance depending on a weight. This Kalman gain is adjusted iteratively so that by estimating the sampled value we get a relatively noise-less value that was used in the Project-app. While a Kalman Filter has theory involving Linear Algebra and matrix multiplication, the Scalar filter used here did not as the matrices reduced to constants. It gave fairly accurate results and some of the basic equations it used were similar to http://bilgin.esme.org/BitsBytes/KalmanFilterforDummies.aspx.

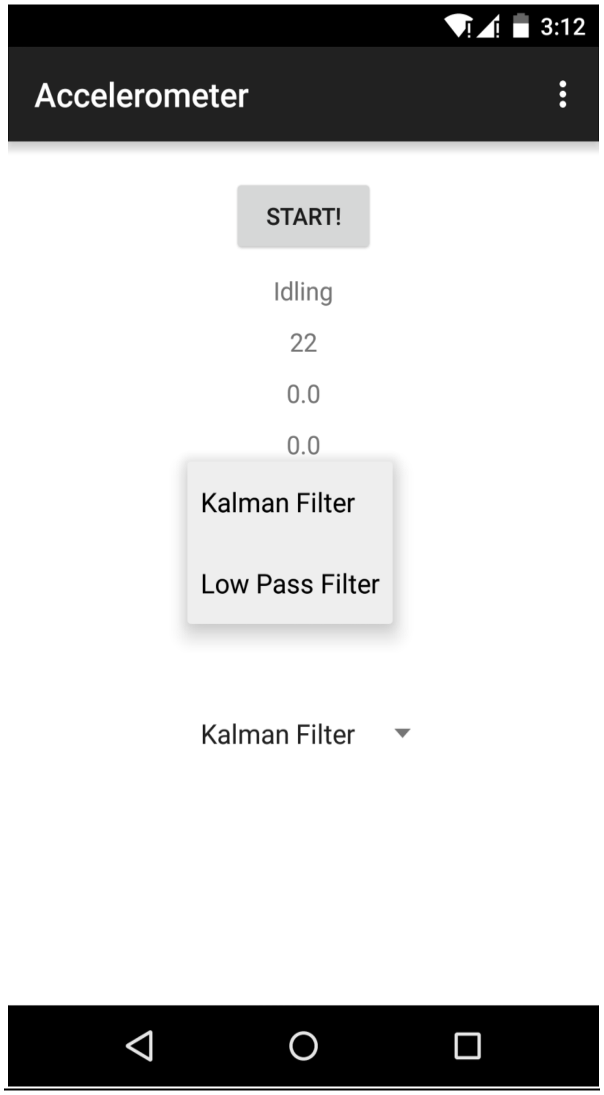
If(k==0) { xk = SensorManager.GRAVITY_EARTH;   Pk = 1; }

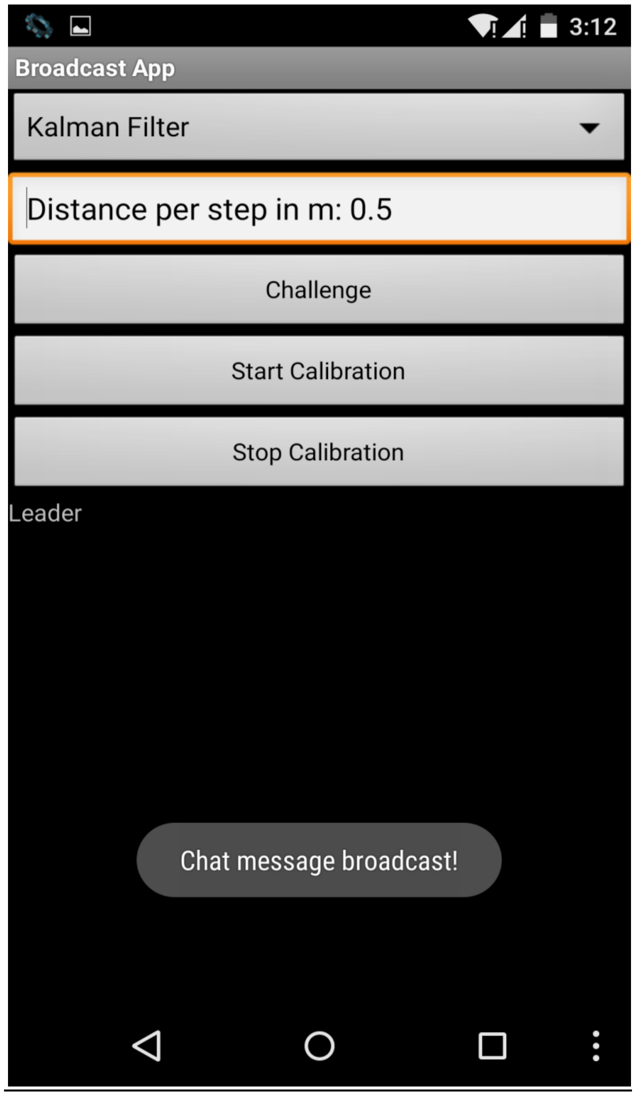
Else { Kk = Pk / (Pk + R);   xk = xk + Kk*(measuredValue - xk);   Pk = (1 - Kk)*Pk; }

Networking and Messaging Lifecycle

This feature was similar to the one used in Programming Assignment 3. The DTN forwarding layer protocol was used which makes it capable of multi-hop (and programmer doesn't need to code for neighbor discovery). In our app, single-hop was sufficient as a typical gym might not need more than one access point or that users might not really be more than a few meters apart (nor out of visible range) for their challenge (though will work in multi-hop mode too, just that scores might not be the most recent in real time). The app broadcasts invites using DTN and creates listeners for receiving replies. This is done periodically and the game status (basically the step count of each player) is sent out by the leader, which was the user who initiated the challenge. The person leading the step count/distance is bubbled to the top of the leader board and this is shown to the users on each device.

Screenshots:





Work share: ~Total lines of Java code = 1200

Prateek Kumar A0123398Y

- Threshold Algorithm for Step Detection
- Kalman Filter/Estimator
- Low Pass Smoothing Average Filter
- Linear Double Integration for distance (not used)

Jabir Shabbir Karachiwala A0123480N

- Broadcast invites over DTN and receive players based on IMEI numbers
- App GUI and Full Messaging Lifecycle
- Estimation using double integration for distance calculation (not used)
- Networking and full Leader board collation.

Testing:

We tested the step detection with the phone in our lower pocket dozens of times and it gives a fairly accurate step count with the Kalman Filter. It also works irrespective of direction, and steps were detected while walking in circles and with turns – because it measures depending on oscillation of the leg which would periodically make one consecutive sample have an absolute difference above a certain threshold (a half step). For step numbers walked/tested like 25, 50, 100, 150, 175, 200 – we got readings almost identical and sometimes off by up to 3-8 steps depending on the total number. The algorithm counts steps alone, not speed so it doesn't measure endurance of the user on the treadmill which would be higher if the user is running. The distance is the product of the steps counted and the distance per step calculated in the calibration stage. During the Demo of this app the TA walked 24 steps (including change of direction) and the Accelerometer app showed a reading of 24 – which was accurate.

We also compared this with a few of the free apps on Google Store named "Pedometer". They gave similar/comparable results to our app with the Kalman Filter. With more smoothing, or the low pass filter (selectable via drop down menu in app) we saw a slightly larger error for step count as we do lose some of the smaller jerks (steps) during the smoothing step so they don't make it over the threshold. Overall the app does step counting reasonably well, if the phone is placed in the lower pocket. The accelerometer data has no "context" around it so there seems to be no way to co-relate the readings with the type of exercise as each may not have a uniquely identifiable pattern of sensor readings, or may not be discernible with ease.

[Thank you/End of Report]