Android pedometer application capable of saving data on the Cloud

MohammadHossein AskariHemmat
University of McGill
Course Instructor: Dr. Z. Zilic

I. Project Description :

This document is submitted for the third assignment of ECSE 649 course. In this assignment we were asked to design an Android Application that is capable of storing the coming data from the mobile sensors into Cloud. We were asked to use BOX [1], which is file sharing and cloud content management service, for this purpose. The energy efficiency of the design had to be taken into account. In my case I used the raw data from Linear Accelerometer to design a pedometer [2] application. This way the amount of data that will be stored in the cloud would be decreased. The Pedometer design will be explained in details later in this document.
My application includes four major blocks as follows:

- A Service, which runs in background, that deals with collecting the data from the accelerometer.
- A Pedometer that uses the raw data from the service and decides whether the coming data matches with the walking pattern or not.
- A Handler that deals with storing the data into the cloud
- A Main activity that wraps all the above blocks

The rest of this document is going to explain all the above blocks in details. In Figure1 the overall application layout is illustrated.

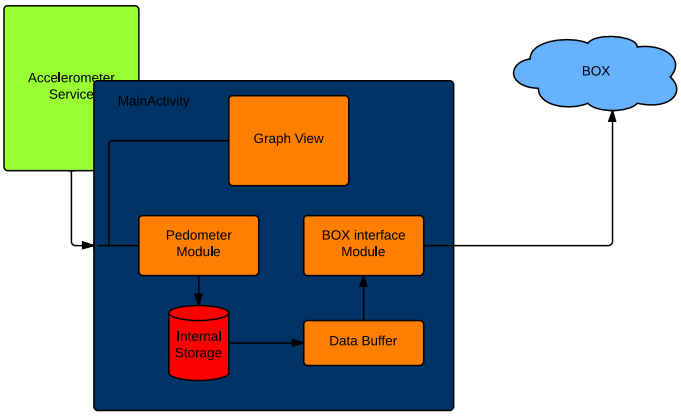

Fig. 1: Application Modules

II. Collecting data from Accelerometer

One of the best ways to collect data from a hardware resource is to use a service in background. The service should be capable of collecting the data from the hardware and then send it to the activities that are binded to the service. Also, the service should be capable of registering to and unregistering from the Hardware when ever it is needed. For instance assume that the battery level is low and the user don't need to collect the data from accelerometer anymore. At the same time he/she needs to use some of the application features regardless of the accelerometer data. It will be very convenient to let user to disable or enable the accelerometer at any time. In my design, user can do the latter by going to the application settings and from there disable or enable the hardware. The settings is illustrated in Figure3. In android simply you can enable or disable a hardware resource by unregistering the

from the listener list.
It is important to know that Android will not let you to power down the Acceleromter sensor. The reason is that there are other applications that are using the accelerometer data and by turning down the accelerometer, the functionality of other applications will be affected.

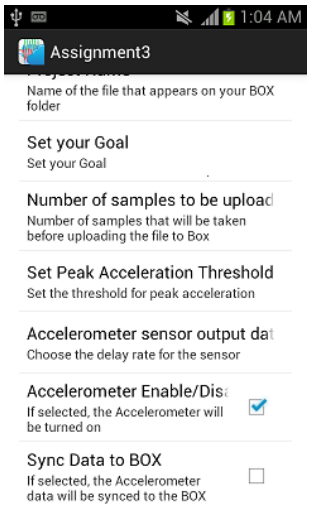

Fig. 2: Disable/Enable Accelerometer sensor

The other way that can do the same thing is to unbind the main activity from the accelerometer service. But the difference is that the serivce will keep collecting data but it would not send them to the activity. Clearly, this is not an efficient way in terms of power consumption.
The other way to save power is by reducing the output data rate of the module. I have also added this feature to my application so that the user could have a number of choice in case that he/she is running out of battery. Android provides four output data rate as illustrated in Figure3 Overall, in order to save power, one of my strategies was to provide the user with some hardware control so his/her decisions can affect the overall power consumption. By default, the acceleromter is turned off and the slowest data rate is selected.

III. Pedometer Module

A pedometer is a device, usually portable and electronic or electromechanical, that counts each step a person takes by detecting the motion of the person's hips [2]. Currently, using 3-Axis accelerometer is a common way to count the number of steps. One of the biggest problem using accelerometer data in a pedometer is filtering the effect of gravity. Usually an extra hardware/software high pass filter is used to filter out the DC

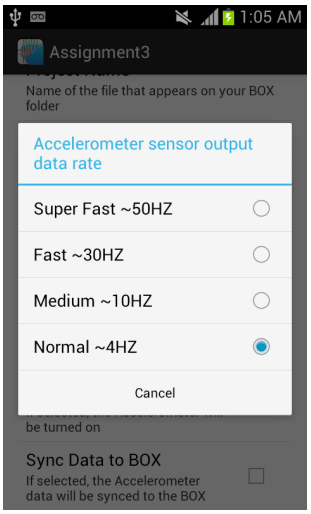

Fig. 3: Acceleromter output data rate

component and based on this result. The other way to solve this problem is to use a linear accelerometer instead. The linear accelerometer can be implemented in hardware or in software. The difference with a common accelerometer is that you do not need any extra piece of Hardware or Software to obtain the accelereation excluding the force of gravity. As a hardware Linear Acceleromter LIS3LV02DL [3] from ST is good example. The Circuit Diagram of this module is illustrated in Figure4

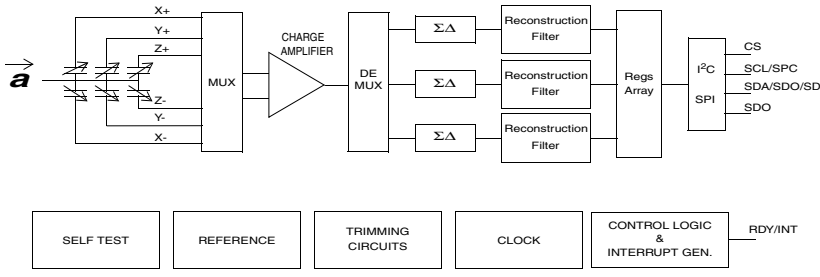

Fig. 4: LIS3LV02DL Circuit Diagram

It can be seen from the Circuit Diagram that a hardware ΣΔ and Reconstruction is used to filter the effect of gravity. As an example of software Linear accelerometer, Google provided this sensor in Android and it can be accessed using *TYPE_LINEAR_ACCELERATION*. In my application, I used Google Linear Acceleration sensor type in my application. A Google Tech Talks of Linear acceleromter can be found in [4].
The raw data from the Linear acceleromter still cannot be used in a pedometer application. In my application I used an FIR filter to smooth out the raw data from the Linear acceleration sensor and

then I feed this data into the pedometer module. To find out how to count a change in acceleromter data as a step or not, We need to understand the pattern of walking. In [5] this pattern has been explained. Based on the model that is presented in latter paper, I came up with a fairly stable algorithm. The algorithm is illustrated in Figure 5.

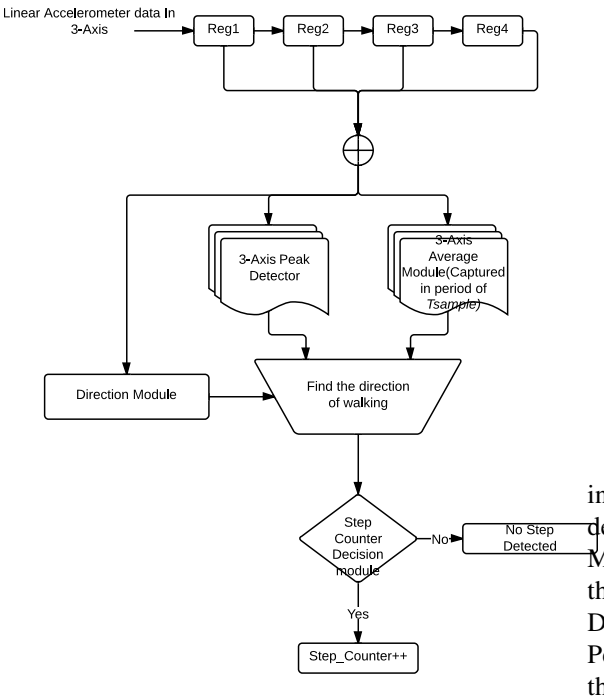

Fig. 5: Pedometer Algorithm

First the acceleromter data is passed threw an FIR filter. Then it passes threw 3 modules as follow:

- Peak Detector Module
- Averaging Module
- Direction Module

All the above modules are working on the 3-Axis filtered acceleration data. Since the rate of data is not constant, the averaging is done based on the data rate that the user has been selected. Considering that the average delay between each step is around 1 second (based on my experience) I used the following formula to find the number of samples that has to be averaged each time:

$$StepSampleRate = StepInterval * DataRate; \quad (1)$$

I set the StepInterval to 1 second and DataRate will be updated each time that the user modifies his/her accelerometer output data rate. The averaging module output and the peak detector output modules are fed to a decision module that decides

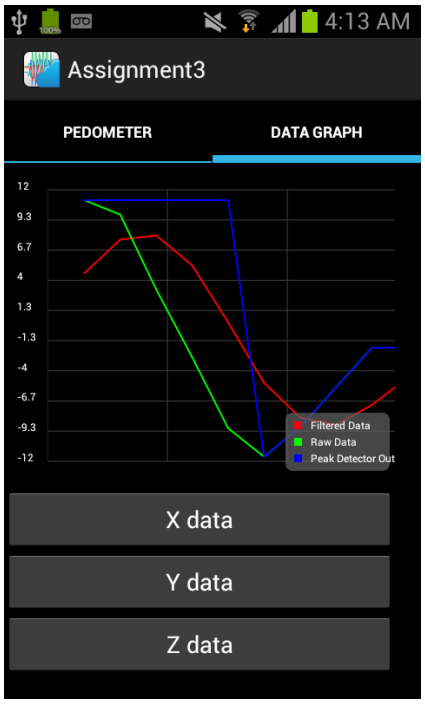

Fig. 6: Output data from the Pedometer algorithm blocks

in which direction change has been occurred. This decision is based on the output of the Direction Module. After finding out the direction of change, the corresponding data is fed into the Step Counter Decision Module. In this module, based on the Peak acceleration threshold, it is decided whether the input data should be count as a step or not. A sample data output from all the major blocks is illustrated in Figure6. This figure was captured from the *DATAGRAPH* in the application Based on my experience, the algorithm works best by setting the Data rate to 30HZ and setting the Peak acceleration data to around 1.2. These parameter can be modified in the application's settings. Also it is available for the user to set a Goal for the number of steps. The Pedometer graphical user interface is illustrated in Figure7.

IV. BOX Interface Module

In order to process/store the data in it is very practical to use a cloud service. In this assignment we used BOX as a cloud storage service. BOX platform supports many other platfroms such as: Android, iOS, Windows and etc. Since we are creating an Android application, we need to download and install BOX Android SDk on the ADT. Here is a link to the SDK in Github: https://github.com/box/box\-android\-sdk\-v2. All the infromation for setting the BOX Android SDk is covered.

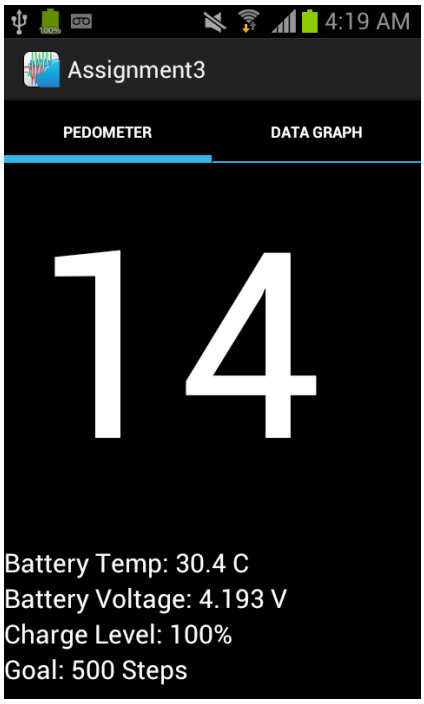

Fig. 7: Pedometer GUI

In order to interact with BOX platform, first we need to create an account in BOX. BOX uses OAuth 2.0 to authenticate and authorize the access of the users to the their account. Applications needs to be aware of three parameters in order to use OAuth: h

- Client ID
- Client Secret
- Redirect Url

Before accessing to BOX using OAuth, its needed to create a Box Application. The later parameters are set in this stage. We need to them in the OAuth parameters section. The Client ID and Client secret ID are generated automatically. But the Redirect url has to set manually. The redirect URI is the URL within your application that will receive the OAuth2 credentials [1]. For most application http://localhost can be set as the Redirect Url parameter. After creating the application in the BOX account, now we need to copy and store these information into the BOX client parameters in our Android application.

To access our Application in BOX, first we need to authenticate our access. This is simply done by starting an OAuthActivity in Android and then call the suitable method based on the response from BOX. Here is Pseudo code:

```
public void StartAuth(){
Intent intent =
    OAuthActivity.createOAuthActivityIntent(CLIENT
    ID,SECRET_ID,url);
this.startActivityForResult(intent,AUTH_REQUEST);
}

// After responding BOX to our
    request:
private void onAuthenticated(int
    resultCode, Intent data) {
     if (Activity.RESULT_OK !=
         resultCode) {
        authentication = false;
              }
              else {
                 BoxAndroidClient
                     client = new
                     BoxAndroidClient(BoxClient.CLIENT_ID,
                     BoxClient.CLIENT_SECRET,
                     null, null);
                 client.authenticate(oauth);
                 if (client == null) {
                    authentication =
                        false;
                 }
                 else {
                    ((BoxClient)getApplication()).setClient
                    authentication =
                        true;
                 }
              }
          }
```

If the authentication was successful, we will able to perform Upload, Download and other methods that are provided by the BOX platform. In my application, authentication will be saved after the first successful access. If the user exits the application, the authentication will be deleted. If he/she preferred to run the application in the background, no authentication will be asked after his/her return. Also it is available to use the application without Syncing the data to the BOX. To do so, user need to Uncheck the SynC Data to BOX option in the settings. In order to save power, user can set the*Number of samples to be uploaded* parameter in the settings menu. This parameter determines how frequent data should be uploaded into the BOX. The minimum value for this parameter is 500 samples which will give only 1 second before each data uploading. By setting this parameter to 500, the power consumption will increase very fast. The default value is 2000. Also to save power, only the number of steps are sent to be stored. Figure 8 shows the setting of this option: The name of file that will be saved into the BOX can also be defined. To avoid losing data while Syncing data to BOX, I Used a File buffer. So the pedometer save its data to a buffer and the BOX interface reads it only if its available.

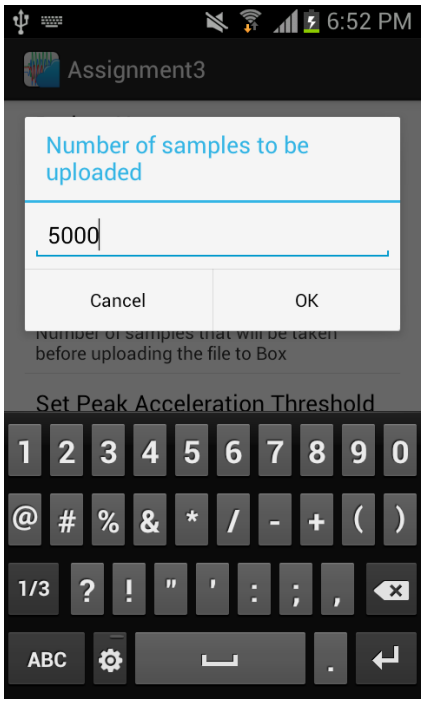

Fig. 8: setting the Number of samples to be uploaded

V. Main Activity

All of the above modules Plus the DataGraph modules are controlled by the main activity. Main activity is responsible to save/delet the files from the external storage device. It is also responsible to save the authentication and ask for re-authentication if its needed.

VI. Conclusion

in this assignment we learned how to use a cloud service to save our data from our android application. We were asked to try to design the most efficient application in terms of power consumption. My application run for 11 hours non-stop in the most power consuming settings which is uploading data every 1s and set the Linear acceleromter data rate to 50HZ.

References

[1] https://app.box.com/
[2] http://en.wikipedia.org/wiki/Pedometer
[3] http://www.st.com/web/catalog/sense_power/FM89/SC444/PF127514?referrer=70032480
[4] https://www.youtube.com/watch?v=C7JQ7Rpwn2k&feature=player_detailpage#t=1349
[5] N.Zhao, "Full-Featured Pedometer Design Realized with 3-Axis Digital Accelerometer",Analog Dialogue 44-06, June (2010)