Detecting Snoring to Inform Night-Time Smartphone Duty-Cycle Scheduling

Thomas Phan
Samsung Research America - Silicon Valley
San Jose, CA USA
thomas.phan@samsung.com

***Abstract*—Mobile sensing applications can infer user behavior at the cost of increased battery consumption due to the use of power-hungry sensors. To reduce power consumption, these applications can either cease sensing or perform duty-cycling at night when the user is sleeping; however, this approach works well only if user inactivity can be detected with certainty and with low power. In this paper we explore the viability of detecting human snoring as a strong indicator of user sleep periods and leverage that inference to perform aggressive night-time duty cycling. To detect human snoring, we collected ground-truth data from a sleeping participant and developed a threshold-based model that is able to identify periods of sleeping. This information was used to mark the onset of sleep, which in turn facilitated better duty-cycling and reduced power consumption.**

I. Introduction

Mobile sensing applications running on modern commodity smartphones can infer user behavior through a variety of different sensor modalities. For example, location-based applications leverage the user's current physical position determined via GPS or other wireless trilateration to provide targeted services [3], and social sensing from examining online browsing and media usage can model social behavior [20].

Note that in all cases, continuous sensing is limited by the battery capacity of the smartphone. Indeed, for power-hungry sensors like GPS that can consume over 500 mW, continuous use will deplete a standard smartphone battery within several hours [11]. To alleviate this problem, sensing applications should cease activity when the user is idling, such as during sleeping, by detecting the absence of motion through the accelerometer (e.g. [1] [19] [15] [9] [13]).

However, this optimization still requires the smartphone CPU to be active in order to sample the accelerometer and perform inferencing to detect periods of idling versus activity. In our experiments we found that this accelerometer sensing consumes approximately 194 mW of power, including a baseline of 172 mW for an idle CPU, on our Samsung Nexus S test smartphone. To reduce this power consumption even further, sensing applications systems can perform fixed or adaptive duty-cycling by allowing the smartphone CPU to go to sleep and then periodically waking up to perform sensing. Duty-cycle scheduling is a tradeoff, where longer periodicities will reduce power but increase the likelihood of missing events. In the case of duty-cycling while the user is asleep, a long duty cycle may miss when the user wakes up.

An advantage could potentially be attained if the smartphone were able to detect that the user was *actually sleeping*. If this were the case, then there would be no need to perform accelerometer sensing to detect idling, and the phone's CPU could be placed inactive for several hours, depending on the amount of sleep that the user tends to demonstrate. This advantage is especially important if the user's phone is not charging overnight. Furthermore, if the system could predict when the user is due to wake up, it could facilitate a variety of Ubicomp / Internet-of-Things scenarios, such as preemptively adjusting the household thermostat, starting a coffee maker, or downloading and formatting news and other content.

In this paper we explore the viability of detecting human snoring as a strong indicator of user sleep periods and leverage that inference to perform better night-time duty cycling. To detect human snoring, we collected ground-truth data from a sleeping participant and developed a threshold-based model that is able to identify periods of sleeping with nearly 80% accuracy. This information was used to mark the onset of sleeping, which in turn allowed us to better predict when the user will wake up. Our results show that an aggressive duty-cycling schedule based on these sleep factors can enable a reduction of consumed power of over 88% relative to an accelerometer-only approach over our data set.

The rest of this paper is organized as follows. We discuss our algorithm to detect snoring in Section II, show how we apply this detection to perform night-time scheduling in Section III, and conclude in Section IV.

II. Snoring Detection

Snoring is typically a result of a respiratory blockage and can be related to the serious affliction of sleep apnea [18]. A sizable portion of the human population snores (44% of men and 28% of women [4]), and for this group, we can apply our approach to improve smartphone duty-cycling.

To detect snoring, medical clinicians use polysomnography, a procedure that requires patients to sleep overnight in a laboratory with wired connections to measurement devices [16]. Recent clinical work has also looked at detecting snoring by analyzing recorded audio of the snoring sound using desktop computing. The work in [5] uses mel-frequency cepstral coefficients in a manner similar to automatic speech recognition [8]. [2] computes the energy in frequency-domain sub-bands, performs principal component analysis on the feature space,

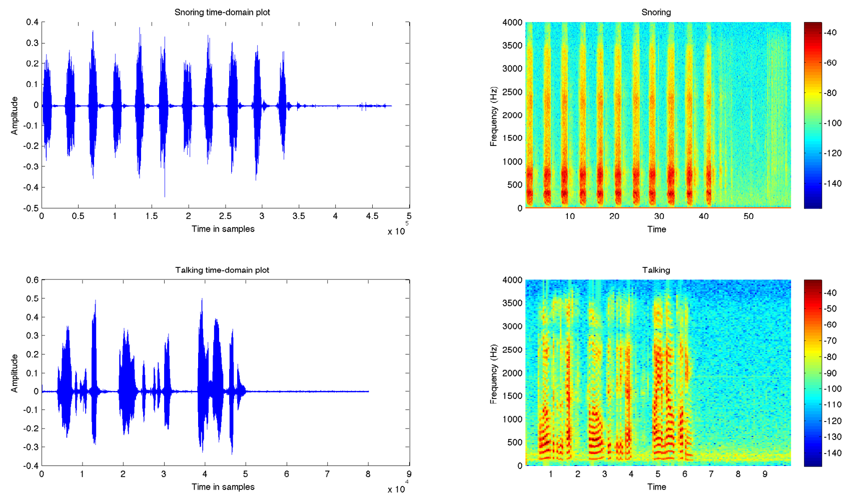

Fig. 1. Time-domain signal plots are shown on the left half of the figure, and frequency-domain spectrograms are on the right. The top half shows data for a 60-second recording of snoring. The lower half shows a 10-second recording of "hello this is a test, hello this is a test, I am now talking." In the snoring data, note that there are 11 clearly-discernible snores in the recording and that the region of high-magnitude density is concentrated below 1000 Hz.

and applies linear regression for classification. [16] generates a Hidden Markov Model to model respiratory cycles.

Our main goal is not to perform a clinical assessment of sleep disorders; instead, we took a simpler path that could be effectively implemented on a smartphone.

Our approach to identifying snoring stems from insight gained by observing spectrograms of snoring and speech from a participant. (We found that our colleagues and friends were understandably unwilling to collect night-time audio data; in the future we will look to collaborate with a clinic to collect more data.) These sounds were captured on a Samsung Nexus S smartphone using an audio recording application sampling at 44100 Hz and producing 16-bit mono data into a PCM file. The snoring was recorded while the participant was sleeping at night in a quiet bedroom, while the speech was captured during the daytime with no ambient noise.

Figure 1 shows time-domain sample plots in the left column and spectrograms (up to 4000 Hz) in the right column. The top row shows the data for a 60-second snoring clip, while the bottom row shows the data for a 10-second speech clip.

There are several important observations. First, there are clearly 11 snores captured in the recording, with the 11 snores visible in the spectrogram. Second, and more importantly, the snoring has high magnitude in the lower frequencies; from close observations, there is a demarcation point at 1000 Hz where the frequency-domain magnitude is most strong, and above this point the magnitude appears weaker. On the other hand, in the speech the magnitude in the frequency domain appears to be spread out further across the frequency spectrum.

This insight led to the following algorithm that uses core signal processing [12]. We used the Android audio-recording API to sample the microphone at f_s = 44100 Hz to continuously fill a 512-sample buffer with 50% overlap and an applied Hamming window. We then ran an FFT on each window, resulting in a frequency-domain magnitude frequency response that we then partitioned into sub-bands of "low" and "high". For this participant, we set the cutoff between "low" and "high" frequencies for the purposes of snoring-detection to be 1000 Hz, as described earlier. Note that assessment for other users would have to be automatically personalized through similar training, such as what is done today with speech or handwriting recognition.

To provide a measure of the magnitude density found in the low and high bands, we computed the discrete energy. For any sequence of samples x_i, the discrete energy E_d of the sample window is found by:

$$E_d = \sum_{i=-\infty}^{\infty} |x_i|^2$$

Let E_{low} and E_{high} be the frequency-domain energy in the low and high regions of the frequency spectrum. The low energy ratio LER can then be defined as:

$$LER = \frac{E_{low}}{E_{low} + E_{high}}$$

From testing on the participant, we found that a window of sample data contained snoring if LER exceeded 0.65. To smooth out random classification flapping, we kept a sliding window moving average of the last 50 LER values (corresponding to 50 windows $\times$ 512 samples/window $\times$ 1 second/44100 samples = 0.58 seconds).

If this moving average exceeded 0.65 for the given window, then it is classified as "snoring"; otherwise it is classified as "non-snoring". A side-effect of using a moving average is that real-time reaction is delayed. Some windows may be misclassified as snoring immediately after exiting a snoring region. Another issue with this algorithm is that it is dependent on the magnitude values, which in turn are dependent on the volume gain of the recordings, and this gain perceived by the microphone is a function of the distance between the microphone and the source. As a result, the algorithm will not work well unless the microphone analogue-to-digital process is able to pick up sounds clearly.

To evaluate our algorithm, we recorded ground-truth microphone data from the participant while he was sleeping over two nights, where the smartphone was placed next to his pillow approximately 12 inches away from his head. From each recording we created 40 segments, each 5 seconds long, of snoring and also non-snoring (which was silence). Applying our algorithm on the segments, we found that it produced an accuracy of 77.5%. Recent research in the area of audio-only snoring recognition using offline desktop computing shows state-of-the-art accuracy of 86% to 100% [16]. For the purpose of identify snoring on the smartphone to enable duty-cycle scheduling, we were generally satisfied with our results. In the future we will look to apply more advanced training and modeling to improve our accuracy. One problematic area where we plan to explore is the false-negative misclassification of various phases of a snore. A snore comprises inhalation and exhalation, and our system generally classified exhalations correctly but not inhalations.

III. Using snores to improve night-time scheduling

Given the snoring detection component described in the last section, we then looked to improve smartphone duty-cycle scheduling at night by exploiting exhibited sleeping behavior. Specifically, we leverage the snoring detection in three ways: (1) the detection of first snoring allows us to define the start of user sleep, which in turn allows us to then disable the detection mechanism and to better measure the duration of sleep; (2) the measured duration of sleep allows us to better predict when the user will wake up; and (3) at the time that the user is expected to wake up, we can enable snoring detection again to see if the user is still asleep, and if this is the case, then we can continue to sense intermittently.

With the user's state distributed as a broadcast Android Intent message, applications can then better schedule their own duty-cycling to conserve battery power while the user is sleeping. In addition to apps that use power-hungry sensors like GPS, other apps that continually check online Web sites, such as social networking apps that download updates [14], should likewise either cease their activity completely or maximally lengthen their duty-cycle period.

In the rest of this section we describe in detail our data collection, sleep prediction classification model, and operational algorithm to improve night-time scheduling.

A. Data collection

To build our model, we collected audio data from a male participant using an Android application running on a Nexus S smartphone. In total we accumulated data from 23 nights, all taken between the hours of 11:00 PM and 9:00 AM local time. To collect data, the participant started the application's audio recording and then placed the phone next to his pillow. The user then went to sleep and woke up without the aid of any external stimulus such as an alarm. We instructed the participant to deactivate the audio recording as soon as he woke up, so we took the end of the sleep to be the time that the user stopped the application.

B. Improving night-time duty-cycling

We first consider an accelerometer-only approach to detecting when the user wakes up [1][19]. If significant motion is detected (either through a threshold-based model or a classification model), then the system can conclude that the user is awake and active. One problem with this approach is that if the user places the smartphone on his bed, then any inevitable and potentially frequent body movement may inadvertently trigger an awakening inference, and so opportunities to reduce power due to idling may be rare and of short duration. Another problem occurs whether the phone is on the bed or on a stable surface: the accelerometer must be continuously sampled to determine the existence or absence of movement.

We can improve upon the accelerometer-only approach to reduce battery usage during the night by using our snoring detection. Consider the screenshots from our audio visualization tool in Figure 2 that show two traces of perceived audio magnitude. We confirmed by manually listening through these two traces that all the data emanates from snoring (as opposed to talking or music).

The plots show a pattern of snoring and silence, which is representative of the human sleep cycle. For example, there are often extended periods of silence, which may be associated with the REM phase of sleep. (In a separate experiment, we recorded data simultaneously from both the microphone and the accelerometer while the phone was strapped to the user's arm. Long periods of completely inert bodily motion, which is indicative of REM, co-occurred with silence, suggesting that for some people, a lack of snoring is correlated with REM.)

In the future we will explore how we can fully leverage these cycles to improve our predictions. However, in our current work we noted that the end of the sleep typically shows a lack of snoring for this participant, and as a result we then looked to see if we could exploit this feature to better determine when the user awakens, which in turn can help reduce or eliminate the need for actively duty-cycling sensors.

If we could indeed use snoring detection to achieve this end goal, then presumably the energy consumed from snoring detection, E_{sd}, should be less than the energy consumed from the accelerometer-only detection, E_{acc}. Let P_{sd}, P_{acc}, T_{sd}, and T_{acc} be the average power draw from snoring detection, the average power draw from the accelerometer-only detection, the total time that the snoring detection is active, and the total time that the accelerometer-only detection is active, respectively. From the basic principal of $Energy\ (Joules) = Power\ (Watts) \times Time\ (seconds)$, it can be seen that $E_{sd} < E_{acc}$ if $\frac{T_{sd}}{T_{acc}} < \frac{P_{acc}}{P_{sd}}$.

We measured average power using a Monsoon Solutions Power Monitor, a hardware power meter that directly provides electricity to a phone through connected leads and measures

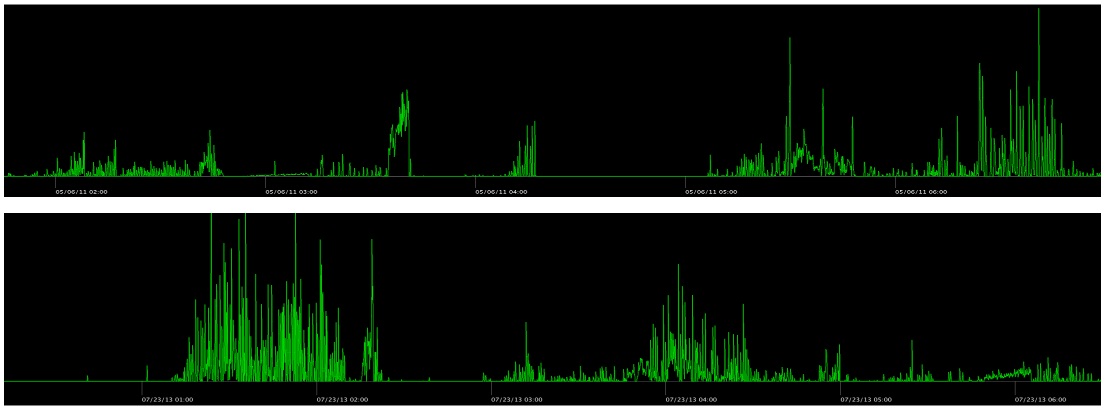

Fig. 2. Screenshots of our audio visualization tool that plots perceived audio magnitude as a time series. Note that the tail end of the series just before the user wakes up is marked by recessed snoring.

the resulting power draw. To perform measurements accurately, we put the phone into "airplane" mode, turned off the cellular radio and GPS, and killed unneeded background processes. We found that the average power consumption for the accelerometer and light computation was P_{acc} = 194 mW, while the power for snoring detection (which includes microphone sampling, running FFT, and the computation discussed in Section II) was P_{sd} = 303 mW. As a result, an operational algorithm must limit the time that the snoring detection is active to be $\frac{T_{sd}}{T_{acc}} < 0.64$.

For an accelerometer-only algorithm, the accelerometer performs sensing throughout the night, so T_{acc} is the total amount of time that the user is asleep.

To determine the time T_{sd}, we first identify three phases of sleep that help inform how we leverage snoring detection.

- Phase 1: The user is preparing for sleep but has not yet fallen asleep.
- Phase 2: The user is asleep. The needed distinction between Phase 1 and Phase 2 will be discussed shortly.
- Phase 3: The user will awaken within the next w minutes. We set w to be 10 since that seemed most appropriate for allowing any inactive sensing applications to begin sensing again. Further, other Ubicomp or social networking applications can use this lead time to perform any needed operations such as checking for online updates.

At first glance, it seems that Phase 1 does not need to be considered since what we really want is the duration of Phase 2. However, consider an estimate of Phase 2, where computing a mean or median duration from collected data requires us to define the start and end of sleeping. As mentioned previously, we determined the end of sleep to be the time that the user deactivates the audio recording application. To compute the start time, one could follow analogously and use the time that the participant started recording. The ensuing problem is that it is not known when the user actually falls asleep even though he is in bed. As a proxy for this event, we use the time of the first recorded snore as the beginning. Our collected data showed that on average the duration between the start of audio recording and the first snore was 32 minutes with a standard deviation of 45 minutes, where this wide deviation was due to outliers where the first snore did not occur until up to 91 minutes after the start of recording.

Once we defined the start of sleep to be the onset of snoring, we were able to compute tighter bounds on the average amount of time that the user sleeps. In the absence of using snoring detection and instead relying on the beginning of the audio recording as the start, we found an average sleep time of $\mu =$ 441 minutes (7 hours, 11 minutes) with a standard deviation of $\sigma = 59$ minutes. With the start defined by snoring, the average was 409 minutes with a standard deviation of 31 minutes. Stated another way using the coefficient of variation $c_v = \frac{\sigma}{\mu}$, not using snoring detection produced $c_v = 0.134$ while using snoring detection produced a tighter $c_v = 0.076$.

While using the average time is a good approximation for how long the user remains asleep, there is still room for improvement because it does not consider the state of the user during the sleep. Instead, we compute our estimate of Phase 2 through Bayesian reasoning that produces a posterior probability over two classes, namely whether or not the user will wake up within the next w = 10 minutes, given two features: (1) the elapsed time $t_{elapsed}$ from the onset of sleep (determined by the start of snoring) and (2) whether or not the user is currently snoring at $t_{elapsed}$. To compute this posterior distribution, we built a Naive Bayes classifier trained on the collected audio data. Following machine learning best practices, we created an equal number of training instances for each of the two classes [7], resulting in equal prior probabilities. When we fix the feature of snoring to be false, we could then sweep through increasing values of $t_{elapsed}$ such that the posterior probability of the user waking up in the next w minutes is maximized, where the maximum probability ended up being 94%. We use this value of $t_{elapsed}$ as our estimate of the duration of Phase 2.

Since we must demarcate Phase 1 from Phase 2 in order to achieve a tighter estimate of the latter, our snoring detection algorithm must be fully active during Phase 1 in order to detect snoring. *The snoring detection can then be deactivated throughout Phase 2.* In our implementation on Android, we used a CPU wakelock during Phase 1 so that the operating system would not kill the snoring detection. Once Phase 2 started, we released the wakelock to let the phone go to sleep and set an OS alarm to wake later after Phase 2 presumably ended, where the time of the alarm is computed from our estimate of Phase 2, as described above.

At the start of Phase 3, the user is presumably prepared to awake in the next w minutes; however, from our logs, there was still duration variability. To treat this problem, we perform snoring detection again for 30 seconds when Phase 3 starts. If no snoring is detected, we proceed with the assumption that, indeed, the user will awaken in w minutes. (We can then broadcast this information to any interested applications.) On the other hand, if snoring is detected, we re-enter Phase 2 for another 5 minutes and repeat the entrance into Phase 3.

C. Results

We evaluated our operational algorithm from the previous subsection against the accelerometer-only approach by emulating their execution on our trace data. The results are summarized in Table I.

	Snoring Detection	Accelerometer-Only
Power (mW)	303	194
Time active (sec)	1980	26240
Total energy (J)	599.94	5090.56

TABLE I
AVERAGED RESULTS OVER TRACE DATA.

On average, the user slept 26240 seconds. The accelerometer-only approach kept the CPU active the entire time. On the other hand, the snoring detection algorithm kept the microphone sensing and processing active only during the portions of Phase 1 (which averaged 32 minutes) and the onset of Phase 3. In this latter phase, snoring detection had to be activated twice on average, where the second (and any subsequent) activation was due to the user sleeping longer than the estimate duration calculated for Phase 2. The ratio of $\frac{T_{sd}}{T_{acc}}$ was 0.075, which was far below the value of 0.64 discussed earlier as a necessary condition for the snoring detection algorithm to consume less energy. Indeed, our algorithm resulted in an 88.21% reduction in energy versus the accelerometer-only approach over our collected data set.

IV. Conclusion and Future Work

We considered the domain of modern smartphone applications, where battery can be conserved during the night by disabling power-hungry sensors like GPS or ceasing network activity from social media software. This conservation is especially important if the smartphone is not charging overnight. However, this approach works well only if user inactivity can be detected.

To that end, we explored the viability of detecting human snoring as a strong indicator of user sleep periods and leverage that inference to perform better night-time duty-cycling. To detect human snoring, we developed a threshold-based model that is able to identify periods of sleeping, which was then used to aggressively allow the phone to be inactive. Compared to an accelerometer-only approach, our algorithm produced a reduction of over 88% in consumed energy over our data set.

In the future we look to evaluate our snoring detection with more participants and a wider variety of application usage. We will also look at suppressing our mechanism if the phone is in use, a feature that is helpful if a different person is snoring.

References

[1] F. Abdesslem, A. Philips, and T. Henderson. “Less is More: Energy-Efficient Mobile Sensing with SenseLess,” In *Proceedings of ACM MobiHeld*, 2009.

[2] M. Cavusoglu, M. Kamasak, O. Erogul, T. Ciloglu, T. Serinagaoglu, and T. Akcam. “An efficient method for snore/nonsnore classification of sleep sounds,” *Physiological Measurement*, 28(8), pp. 841-853, 2007.

[3] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. “Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project,” In *Proceedings of ACM MobiCom*, 2000.

[4] L. D’Andrea. “Why do people snore?,” *Scientific American*, April 2004.

[5] W. Duckitt, S. Tuomi, and T. Niesler. “Automatic detection, segmentation, and assessment of snoring from ambient acoustic data,” *Physiological Measurement*, 27(10), pp. 1047-1056, 2006.

[6] S. Fang and R. Zimmerman. “EnAcq: Energy-efficient GPS Trajectory Data Acquisition Based on Improved Map Matching,” In *Proceedings of ACM SIGSPATIAL*, 2011.

[7] V. Garcia, J. Sanchez, R. Mollineda, R. Alejo, and J. Sotoca. “The Class Imbalance Problem in Pattern Classification and Learning,” In *Congreso Espanol de Informatica*, 2007.

[8] D. Jurafsky and D. Martin. Speech and Language Processing, 2nd ed. Pearson Prentice Hall, 2008.

[9] D. Kim, Y. Kim, D. Estrin, and M. Srisastava. “SensLoc: Sensing Everyday Places and Paths using Less Energy,” In *Proceedings of ACM SenSys*, 2010.

[10] M. Kjaergaard, J. Langdal, T. Godsk, and T. Toftkjaer. “EnTracked: Energy-Efficient Robust Position Tracking for Mobile Devices,” In *Proceedings of ACM MobiSys*, 2009.

[11] J. Liu, B. Priyantha, T. Hart, H. Ramos, A. Loureiro, and Q. Wang. “Energy Efficient GPS Sensing with Cloud Offloading,” In *Proceedings of ACM SenSys*, 2012.

[12] A. Oppenheim and R. Schafer. Discrete-Time Signal Processing, 3rd ed. Prentice Hall, 2009.

[13] T. Oshin, S. Poslad, and A. Ma. “Improving the Energy-Efficiency of GPS-based Location Sensing Smartphone Applications,” In *Proceedings of IEEE TrustCom*, 2012.

[14] A. Pathak, C. Hu, and M. Zhang. “Where is the Energy Spent Inside My App? Fine Grained Energy Accounting on Smartphones using Eprof,” In *Proceedings of EuroSys*, 2012.

[15] I. Shafer and M. Chang. “Movement Detection for Power-Efficient Smartphone WLAN Localization,” In *Proceedings of ACM MSWiM*, 2010.

[16] B. Snider and A. Kain. “Automatic Classification of Breathing Sounds During Sleep,” In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[17] Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishamachari, and N. Sadeh. “A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition,” In *Proceedings of ACM MobiSys*, 2009.

[18] WebMD website. “Snoring Causes and Health Risks Associated With Snoring,” www.webmd.com/sleep-disorders/guide/snoring-basics, retrieved August 25, 2013.

[19] Z. Zhuang, K.-H. Kim, and J. Singh. “Improving Energy Efficiency of Location Sensing on Smartphones,” In *Proceedings of MobiSys*, 2010.

[20] S. Zulkefly and R. Baharudin. “Mobile Phone Use Amongst Students in a University in Malaysia: Its Correlates and Relationship to Psychological Health,” *European Journal of Scientific Research*, 27(2), 2009.