# Android operating system intermittently delays background access to sensor data.

**Boris Gafurov**
Thinksense Inc.
McLean, 8200 Greensboro Dr, Suite 700,
McLean, VA 22102, USA
bgafurov@thinksense.com

**Angelos Stravou**
Thinksense Inc.
McLean, 8200 Greensboro Dr, Suite 700,
McLean, VA 22102, USA
angelos@thinksense.com

***Abstract*—Since mobile devices were equipped with various sensors, researchers are trying to utilize them in every aspect possible from entertainment to medical fields. Medical use of sensor data aimed at behavior recognition to evaluate owner's wellbeing is of most importance since it can provide early detection or valuable information in between doctor's visits. To make high reliability prediction, the data validity must be of no concerns. This may be a difficult task when continuous data collection is needed because all modern mobile devices built to conserve battery by restricting applications operation, especially when it is brought into background. We developed an app that continuously monitors one or more sensors in the background and recorded raw and system sensor data timestamps at various battery levels and CPU loads. Average access times were very consistent, however, intermittent large (up to 30% of polling period) delays in sensor data access had been observed that had a potential to cause missing data at higher sampling rates. Since, number of delays observed did not depend on the conditions used, it showed that sensor access must have been interrupted by the operating system high level functions. Testing on Google Pixel 6a and Samsung S22 devices revealed safe sampling at 40Hz and 16Hz, respectively. As a result, we propose preliminary profiling on each manufacture/device, if higher sampling rates of the continuous sensor data acquisition are desirable.**

***Keywords—mobile devices, sensors data, power management.***

## I. Introduction

Modern smart devices, iPhones/Pads/Pods, Tablets etc., are all equipped with array of sensors, motion, magnetic, temperature, acoustic etc., that provide useful data about the device conditions and user behavior. That data can be used in many applications that could benefit an end user's wellbeing and experience and also care providers that can make predictions based on the data. To make a quality prediction one needs reliable data sets to do so. However, smart device manufactures main advertisement point is to make device battery use as conservative as possible. This fact just goes against data reliability. How limiting are these restrictions is hard to estimate without actual trials and errors for each manufacturer device.

Collecting and using sensor data is very hot topic in many research projects. Majority of them are aimed at detecting owners' behavior or activity recognition. These studies found valuable information on how many sensors, which ones, and in what combinations work best [1]. However, how quality of the data itself impacts it, what conditions affect it or what type of errors one has to expect has not been fully addressed. It has been found, so far, that there are clear deviations in sensor reading [2] that appear on its own, or as a natural drift [3], or temperature effect and unwanted vibrations [4] can also produce incorrect readings.

Operating system itself and energy conservation mechanisms built-into it may as well play a role in sensor data reliability and deserves careful investigation. This can be crucial, especially for the processes that run in the background mode, since OS will always try to shut them down. That is why this study was aimed at investigating how reliable the sensor data is when collecting continuously from the application that is designed for a background and very long data acquisition. This will provide us with the insights of the types of errors researchers have to deal with and may explain already reported discrepancies and deviations in sensor data.

## II. Methods

Two manufactures were chosen for this study Google and Samsung. The latest smartphones with Android 11 operating system from both manufactures, i.e., Pixel 6a and S22 were used. Devices were stripped of all unnecessary application as much as possible. Both devices were able to stay on one charge for at least 48 hours while recording single sensor data at 5Hz sampling frequency.

### *A. Application design*

Our in-house application developed in Android Studio (Electric Eel, 2022.1.1) was made to capture and store to the local database data from up to two sensors: accelerometer and gyroscope. Application was intended to work in the background, so it had to have Wake Lock enabled and Battery Optimization disabled. Only after accomplishing the later features, we were able to achieve continuous 5Hz sampling rate of the sensor data acquisition when app was in the background (in foreground it was able to do that without these features). To assess operating system impact on the sensor data acquisition we stored two timestamps along with the sensor metrics (x, y, z of accelerations and/or rotational speeds). The first timestamp was based on a "raw" sensor time that comes with the sensor data which reflects the time when sensor data changed, counted from the last time device was rebooted. We converted this raw time into the "wall clock" time by subtracting it from the elapsed time (i.e. time from the last reboot) at the time when the app receives new data from sensor (this calculation produces always negative number, that essentially shows the delay from the time sensor data changed to when app began to process it). All we need is to subtract that delay from the current system time to get the "wall clock" time when the sensor data change happened. If this time is equal or greater than 200ms (5Hz sampling rate) counting from the last time sensor data was received, we store it into the data base. Along with this time we store the second one which is actual system time when data arrived in the app. This two times will allow us to document if there are delays imposed by the operating system / device's hardware that can interfere with continuous sensor data acquisition.

### *B. Data acquisition conditions*

To investigate if these delays are influenced by different factors, sensor data was collected under several conditions:

100% or 10% charged battery, 100% charged battery at a high CPU load. These conditions will elucidate how battery consumption and its level affect the delays when accessing the sensor data.

Each of these conditions were used when collecting data from single sensor, i.e., either accelerometer or gyroscope, or from both simultaneously. Normally, apps collect data from multiple sensors and these experiments will help see what strain it puts on operating system and impact that may have on the quality of the sensor data.

All above data will be collected in two modes stationary and moving. Sensor data are most of the time used to identify behavior or activity that device owner is engaged in. Thus, device displacements need to be incorporated into our project to reveal what impact it may have on the delays, if any. Device's movement will be done with the help of Elephant robotic arm myCobot 280 using the same trajectory stored in the arm's memory.

Data in each condition would be collected for 5min, that was determined to be the maximum time during which battery percentage does not change under the high CPU load. Middle 3min portion of that will used for further off-line analyses.

### C. *Data and Statistical analyses*

Raw sensor timestamps, when sensor data changes occurred, were recalculated into intervals between them. System timestamps, when sensor changes were assessed by the application, were recalculated into the delays from when the change occurred. Data on the intervals and delays variability is presented on the bar plots figures and expressed as means ± standard deviations. The variability of the mean interval between sensor readings across different acquisition conditions will be analyzed for the significant changes using Excel 2 tailed t-test function. To show significant changes in the distributions of the intervals between sensor readings, data will be analyzed using two sample Z-score test formula written in Excel. 3D sensor data converted into a simple geometrical distance or rather acceleration or rotational speed for gyroscope absolute value would be routinely tested against the intervals between sensor readings for cross correlation to ensure that there are no underlying relationships between them. Cross correlation function (CCF) vs. lag relationships will be calculated with no data wrapping and plotted out in Excel using custom written macros to identify any interplay between the sensors. Throughout the text of this manuscript averages will be expressed as a mean ± standard deviation (SD) in milliseconds followed by number of measurements in parentheses.

## III. Results

### A. *Google Pixel 6a phone*

*Accelerometer study* showed great stability in "raw" timestamps, e.g., 100% charged stationary device had average interval between sensor data 200.8±0.9 (898). That was very close to our preprogrammed polling rate of 200ms but constantly delayed 1-2ms. The rest of the conditions showed similar means closely, within 0.5ms, spaced from each other. Statistical tests came back significant on all values, we assume because of very great number of measurements and very tight distributions. We omitted them on the Figure 1, left panel and just have to document that all conditions devised in these sets of experiments may be considered unique and averages though significant, however, had no great absolute differences.

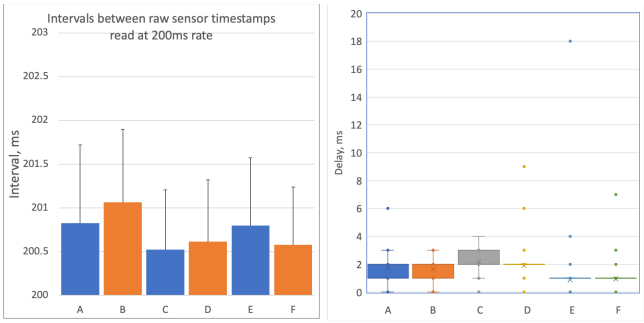


Fig. 1. Intervals and delays between Pixel6a raw accelerometer sensor timestamps. Left panel: A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving. Right panel: Whisker box plots of intervals between the row timestamps for A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving.

Intervals between system timestamps, when application receives sensor data, and "raw" timestamp, that essentially show delays in the processing of sensor data, are presented on the Figure 1, right panel. Average delays vary around 2ms, e.g., the highest for 10% charged stationary device and lowest for 100% charged with high CPU load, were 2.1±0.6 (898) and 0.9±0.7 (898), respectively (Fig.1C,E, right panel). However, that was not of a great concern because one should expect some delay on millisecond scale. The fact that several conditions showed much greater delays, all the way up to 18ms (Fig.1E, right panel), is worth paying attention to because had we been acquiring sensor data at a higher rate, we could have missed those points.

*Gyroscope* sensor data acquisition from Pixel6a, similar to the accelerometer study, showed uniformed "raw" timestamps (Figure 2, left panel) that were very close to polling rate, e.g., 100% charged stationary device had average interval between sensor data 201.0±0.8 (895). Other conditions showed similar, within 0.5ms, means too. Overall, again, they were constantly delayed by 1-2ms from polling rate of 200ms.

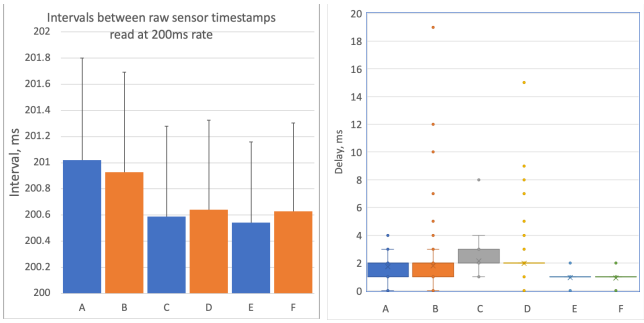


Fig. 2. Intervals and delays between Pixel6a raw gyroscope sensor timestamps. Lefft panel: A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving. Right panel: Whisker box plots of intervals between the row timestamps for A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving.

Delays in the processing of gyroscope sensor data, are presented on the Figure 2, right panel. Average delays, that were remarkably similar to accelerometer, vary around 2ms, e.g., the highest for 10% charged stationary device and lowest for 100% charged with high CPU load, were 2.2±0.7 (897) and 0.9±0.4 (898), respectively (Fig.2C,E, right panel). However, conditions that showed much greater delays, all the way up to 19 and 18ms (Fig.2 B and D, right panel, 100% charged moving and 10% charged moving, respectively) were different from

accelerometer study. The only similarity that can be drawn from them is the movement, that seems to cause more delays.

*Accelerometer and Gyroscope* simultaneous data acquisition produced reminiscent to the single sensor studies. Namely, intervals between "raw" timestamps were very consistent 201.0±1.0ms (Fig.3). Delays analysis (Fig.4) showed similarity with single sensor studies. However, it was clearly noticeable that movement introduced greater number and duration of the delays with 23ms at the peak across all conditions, 100% charged, 10% charged, and 100% charged high CPU load (Fig.4.B1,2, D1,2, and F1,2).

Whisker plot of distributions revealed that longer up to 10ms intervals were again present but to the moderate extent of 15 in total, with 7 for accelerometer and 8 for gyroscope. It seems that these long intervals always appear, however their number depends on sensor type. Moreover, it depends on the number of sensors read, because they appeared more uniformly under the current conditions.

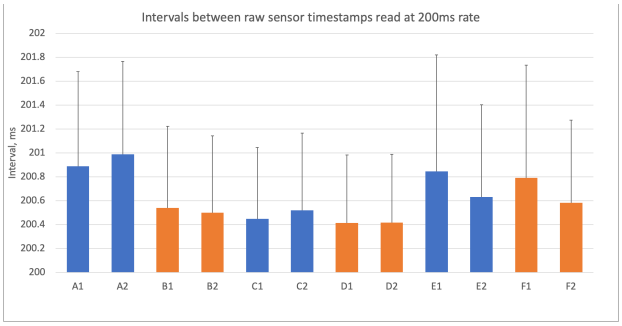


Fig. 3. Intervals between Pixel6a raw timestamps from both sensors are very close to polling rate. A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving. Indexes 1- accelerometer, 2- gyroscope.

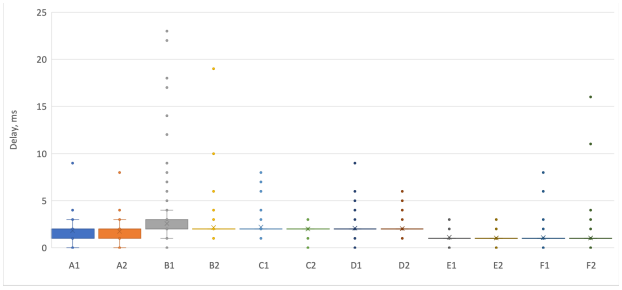


Fig. 4. Delays processing Pixel6a sensor data from both sensors are the highest when devices are moving. Whisker box plots of intervals between the row timestamps for A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving. Indexes 1- accelerometer, 2- gyroscope.

Further cross-correlation analyses (Fig.5) showed consistent lags with R=0.4 in 100% charged and high CPU load when moving devices. 100% charged had lags at 0, 7, 10, and 17ms, when high CPU loaded devices showed -20 and 23ms lags between accelerometer and gyroscope delays. These results hints at the interplays the accelerometer exerts at gyroscope for the most part. In high CPU load, however, gyroscope also has an ability to drive accelerometer delay, that is supported by the negative (-20ms) lag. This synchrony reversal indicate that there is some hardware relationship between these sensors that can contribute to the operating system delays in data processing.


Fig. 5. Movement of Pixel6a introduces cross-correlation between the OS delays in sensor data processing. Cross-correlation between accelerometer and gyroscope delays shows positive peaks when delays were the highest (see Fig.4B,D,F) that appeared when device was moving.

## B. *Samsung S22 phone*

*Accelerometer study,* similar to the Pixel 6a, revealed the intervals between "raw" timesteps were very consistent 201.0±1.0ms, e.g., 100% charged stationary device showed 200.8±0.9(896) average interval (Fig.6A, left panel). Delays imposed by the operating system, in contract to the Pixel 6a phone, were virtually uniform across all the conditions with the peak value of 14ms in 100% charged stationary device (Fig.6A, right panel). This fact strongly suggests that the hardware differences in the devices have a crucial impact on the sensor data processing by the operating system.

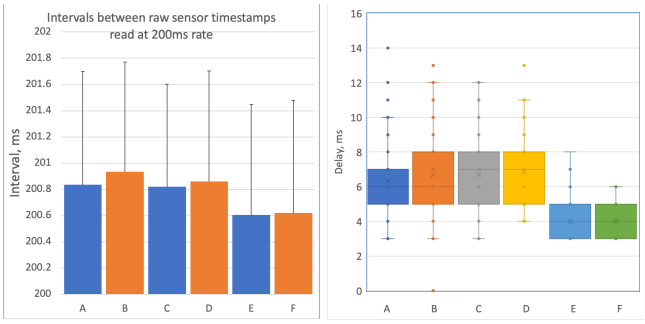


Fig. 6. Intervals and delays between S22 raw accelerometer sensor timestamps are very close to polling rate. Left panel: A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving. Right panel: Whisker box plots of intervals between the row timestamps for A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving.

*Gyroscope study,* again, was similar in the regard to the "raw" intervals between the timestamps. Namely, intervals were very consistent with the polling rate of 200ms and showed approximately 201.0±1.0ms averages across all conditions. E.g., 100% charged stationary device showed 200.8±0.8(896) average interval (Fig.7A, left panel). Delays when accessing gyroscope data (Fig.7, right panel), analogously to the S22 accelerometer, showed uniform character, i.e., they were all up to 14ms with one exception for the 100% charged device that had singular 25ms delay (Fig.7A, right panel). These data were again in the contrast to the Pixel 6a gyroscope delays, where it was documented high impact of the device movement on the delay's magnitude.

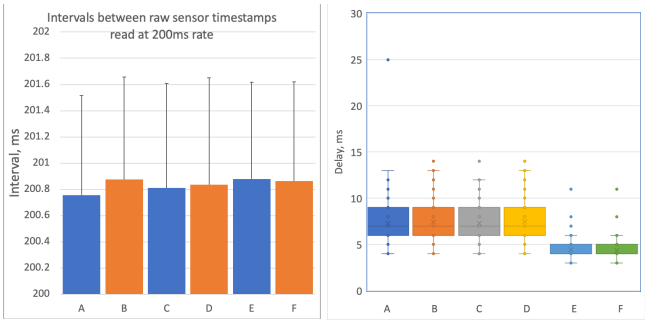


Fig. 7. Intervals and delays between S22 raw gyroscope sensor timestamps. Left panel: A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving. Right panel: Whisker box plots of intervals between the row timestamps for A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving.

*Accelerometer and Gyroscope,* simultaneous sensor data acquisition showed almost no changes when compared with single sensor study (Fig.8). Intervals between "raw" timesteps stayed the same very consistent to 201.0±1.0ms.

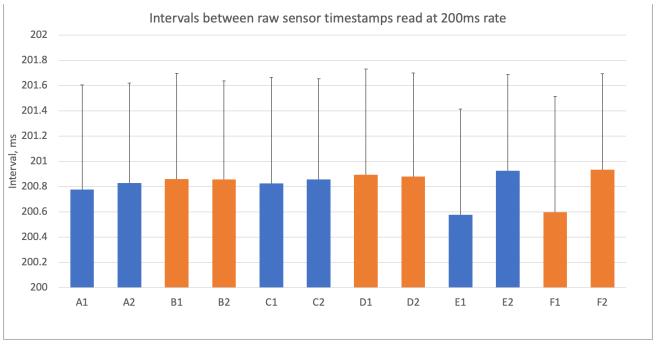


Fig. 8. Intervals between S22 raw timestamps from both sensors are very close to polling rate. A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving. Indexes 1- accelerometer, 2- gyroscope.

Delays, as it can be seen on Figure 9, were uniform once again across all conditions. Although, there was one very high delay of 54ms in accelerometer reading for 100% charged stationary device (Fig.9E1). Other than that, the delays were ranged between 10 and 14ms overall. Also, there were no noticeable impact of the device movement on the delays, as it was documented for Pixel 6a. Cross-correlation analyses did not reveal it either (Fig.10). All lag vs. correlation plots did not show any distinguishable peaks and were generally lower than 0.2. This strongly points at the fact that gyroscope and accelerometer sensors are quite independent hardware units in S22 phone.

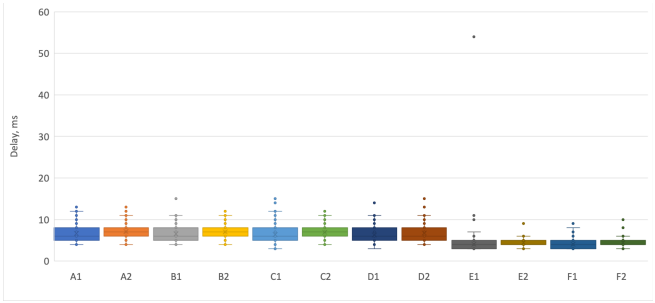


Fig. 9. Delays processing S22 sensor data from gyroscope are similar in all conditions. Whisker box plots of intervals between the row timestamps for A - 100% charged battery, C – 10% charged battery, E – 100% charged battery with CPU at high load when device was stationary. B, D, F – same conditions but when device was moving. Indexes 1- accelerometer, 2- gyroscope.

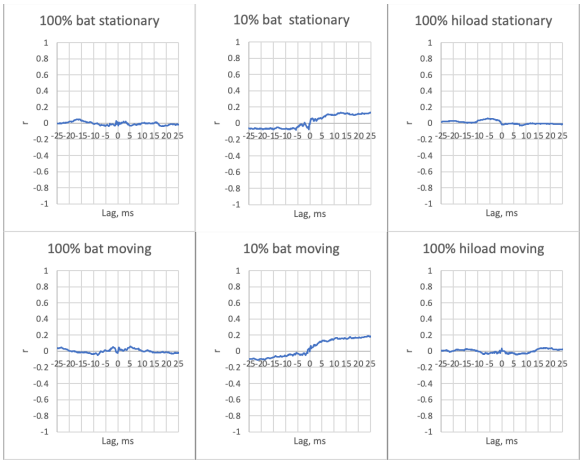


Fig. 10. OS delays in S22 gyroscope and accelerometer sensor data processing show no correlation. Cross-correlation vs lag between accelerometer and gyroscope delays did not show significant peaks and overall was less than r=0.2.

## C. *Stress test of the sensor data acquisition*

Lastly, we decided to test our prediction that there might be data points missing when acquisition is done at the higher sampling rates. We selected 8 always on sensors for these experiments and plotted out the raw delays between the data points. Also, we used only Google Pixel 6a phone, that was 100% charged and moving because that, as it was shown above, had the highest raw intervals between the reading under these conditions. This device has very powerful CPU with 8 cores, so, theoretically, it should be able to handle 8 sensor events simultaneously. In this part we were trying to illustrate intervals between sensor events and determine how close they were to the ideal polling periods determined by the sampling frequency.

*5Hz sampling rate* (200ms polling period) set did not show intervals that could potentially result in missing data (Fig. 11). However, some sensors showed times that were not as close to the time determined by the sampling rate, i.e., 200ms but were delayed by 10-14ms. Among those were linear accelerometer, gravity, magnetic field, and magnetic field uncalibrated sensors. Linear accelerometer showed the most differences with an average delayed by 5ms from 200ms, perhaps because it operates at a lower frequency when compared, e.g., with accelerometer sensor.

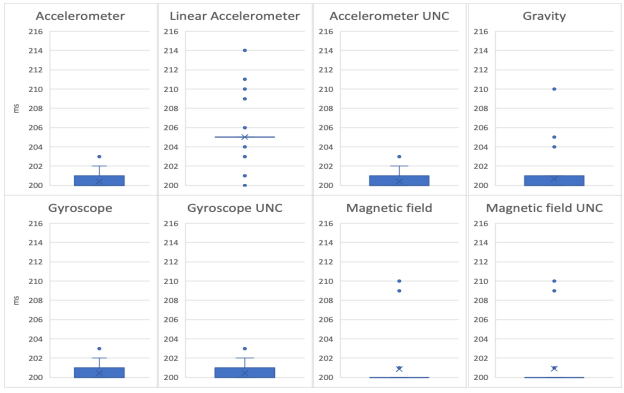


Fig. 11. 5Hz sampling rate. Whisker box plots of intervals between Pixel6a raw timestamps for 8 simultaneously accessed sensors. UNC stands for uncalibrated sensor data.

*20Hz sampling rate* (50ms polling period) was similar to 5Hz sampling rate (Fig. 12). I.e., linear accelerometer was delayed on average from 50ms by 5ms with up to 9ms maximum delays. Magnetic field sensors showed up to 11ms maximum delays in raw between data intervals. Overall, it was still passable to collect from all these sensors at 20Hz (50ms between

points intervals) sampling rate. However, linear accelerometer raised our concerns, because technically approximately every 10th or so point will be absent since it operates at consistently higher/lower (55ms/18Hz) delay/frequency between the points. However, it does not constitute a loss of data but rather systematic lower acquisition frequency that may have an impact on all sensors data validity when taken together.

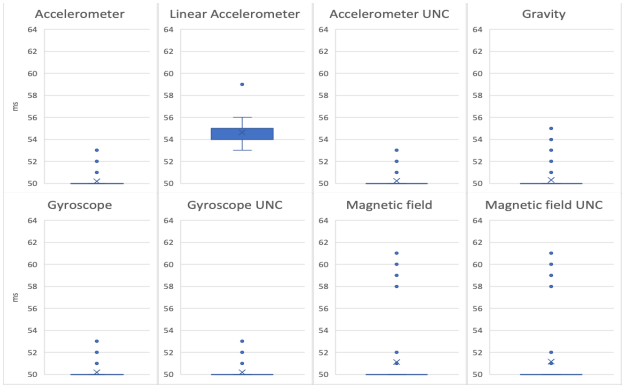


Fig. 12. 20Hz sampling rate. Whisker box plots of intervals between Pixel6a raw timestamps for 8 simultaneously accessed sensors. UNC stands for uncalibrated sensor data.

*50Hz sampling rate* (20ms polling period) case started to show some questionable data collection situations, namely and again, for linear accelerator, and magnetic field sensors (Fig.13). Their raw intervals showed instances where it was 50% or more greater than 50Hz sampling rate imposed minimal timing of 20ms. There is a possibility of those two points appearing back-to-back, thus, creating a missing point in between. Linear accelerometer and gravity sensors also showed consistently greater average interval between the points (27ms and 23ms, respectively). Thus, the later sensors may approximately "miss" every other and every 5th data point, respectively. The bottom line of this observation is that some sensors data is not going to be reasonably synchronized and the degree of uncertainty between them might become too high for some analyses and calculations, e.g., calculation of earth frame from phone frame coordinates.

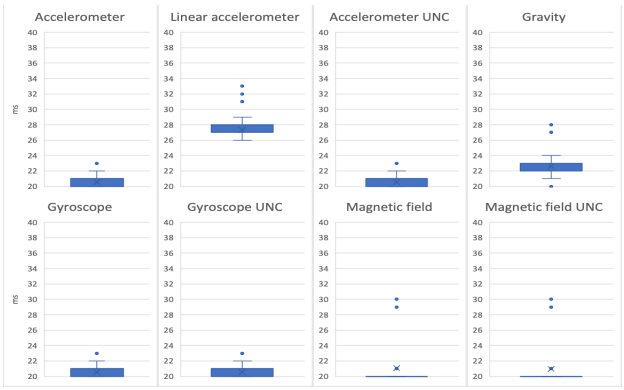


Fig. 13. 50Hz sampling rate. Whisker box plots of intervals between Pixel6a raw timestamps for 8 simultaneously accessed sensors. UNC stands for uncalibrated sensor data.

*100Hz sampling rate* (10ms polling period) case showed even more discrepancies than 50Hz sampling case situation (Fig.14). It has also become apparent that linear accelerometer cannot provide data with shorter than 14ms interval. However, gravity, and magnetic field sensors could but displayed delays up to 10ms, that, at this sampling rate, is detrimental to the data validity.

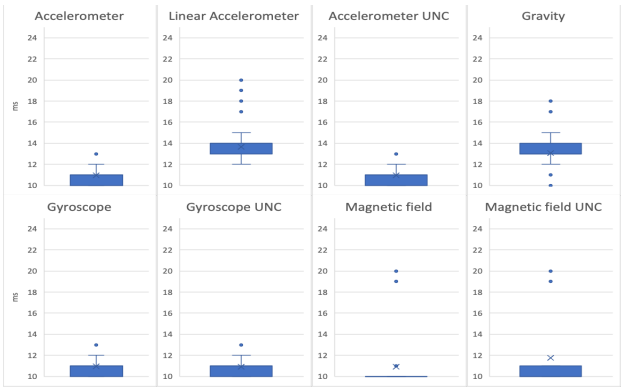


Fig. 14. 100Hz sampling rate. Whisker box plots of intervals between Pixel6a raw timestamps for 8 simultaneously accessed sensors. UNC stands for uncalibrated sensor data.

*200Hz sampling rate* (5ms polling period) case predictably showed even more restrictions on the sensors acquisition frequency (Fig.15). That is, magnetic field sensors cannot operate below 10ms delay between the data points. Gravity sensor still can but displays great delays up to 14ms. Linear accelerometer, analogously to the previous sampling rates, showed great delays up to 24ms with average interval between the point of 14ms. Interestingly enough, accelerometer and gyroscope showed the least impact by the higher acquisition frequencies in all sampling rates tested.

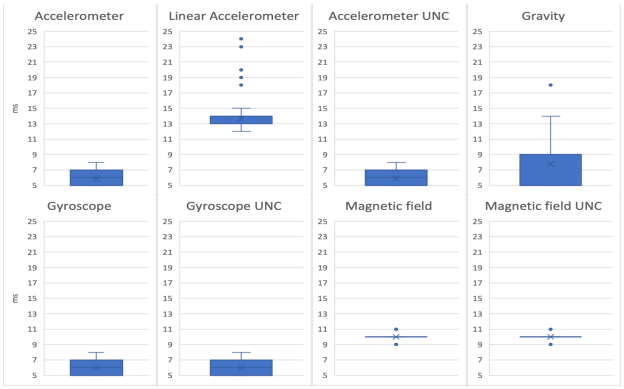


Fig. 15. 200Hz sampling rate. Whisker box plots of intervals between Pixel6a raw timestamps for 8 simultaneously accessed sensors. UNC stands for uncalibrated sensor data.

## IV. Discussion

It is very crucial for the mobile applications that monitor sensors to acquire sensor data accurately and timely, especially in medical research. Especially, it becomes challenging when app is placed into the background by the user, other applications or when device goes into the sleep mode. Manufactures are very strict on saving the battery life to improve user experience and for marketing purposes, so once app is in background or device is in the sleep, all nonessential processes are suspended completely or periodically. For the applications that collect data continuously it could be detrimental. Luckily in Android OS, there is still a way that developers may use to deal with this and ensure uninterrupted data collection from sensors. Apple iOS lacks such a support, unfortunately. However, it appears that even in Android there is still a limitation imposed by the operating system itself where it delays application's behavior.

### *A. Delays*

These delay artifacts may not affect the data that application collects from the sensors, especially the timestamps that come with the sensor information if application does not rely on the system time, but the raw sensor timestamps. The delays will

begin to affect the data being collected when they become comparable with the sampling rate. We collected the data with the intervals between the points up to 200ms, i.e., 5Hz sampling rate. The largest delay that we observed in our data was 54ms (Fig.9E1) on Samsung S22 device and 23ms (Fig.4B1) on Google Pixel 6a. Since there were no gaps in actual data, we can conclude that collecting data as high as 16Hz and 40Hz sampling rate on Samsung and Google phones, respectively, would not introduce any missing data points. If it were to collect faster data points that come to the app, some points might be skipped, if app's behavior had been delayed by the operating system.

Skipping several points may not be detrimental for some tasks that app is intended to do. As well, some tasks do not require high frequency sampling either, say for step detection [5]. However, when every point counts and higher sampling is beneficial, e.g., trajectory tracing, detecting shaking, tremors, convulsions or seizures, these devices may not be suitable, or application has to be adjusted to receive all the data without any gaps because missing points will introduce substantial divergences in the calculated features [6][7]. There are open-source data sets from mobile devices that contain high frequency data with frequencies up to 30Hz and higher [8]. These data review by us, showed very consistent timing without any gaps. However, these data were collected with application in foreground for short periods of times. This may be an option, but some devices do not even allow never go to sleep, and the users will not be able to use their devices during the recording sessions.

### B. *Manufactures*

In our results we noticed that there were distinct differences in the number of delays imposed by the OS between the manufactures, as it has been reported before that there is a great deal of nonhomogeneity between the devices [9]. Namely, Google Pixel 6a showed greater number of delays when device was moving when compared to the other conditions (e.g., Fig.4). Samsung device had more uniformed distribution of the delays between the conditions (e.g., Fig.9).

Moreover, there was very sizable correlation between the sensors in Pixel 6a but only when it was moving (Fig.5). There must be some underlaying mechanisms, perhaps on the hardware level, that increases this unwanted artifact.

It seems that each device needs to be thoroughly tested before setting up an application for sensor data collection. This could be a tall order to devise all the profiles for each device/manufacturer [10]. On the positive note, it appears that single sensor vs multi-sensor data reading produces predictable results. In other words, our simultaneous accelerometer and gyroscope sensors readings were reminiscent of each other. So, the preliminary testing/profiling could be done on the limited number of sensors.

### C. *Acquisition rate limitations*

Sampling rate is crucial to the tasks you want to achieve [11][12][13]. Some sensors cannot function at higher rate, and for some sensors data reading is delayed by the system. However, some sensors, e.g., accelerometer and gyroscope, could handle "virtually" anything in our experiments. These are the factors that has to be accounted for before setting up an app for sensor data acquisition. This finding brings one critical aspect that needs to be monitored. It is the fact that data from sensors may not be well synchronized. This uncertainty between the sensors, e.g., gyroscope data can come from significantly different point of time when compared to accelerometer, has been noticed in Pixel6a. Namely, cross-correlation analysis showed that the synchrony can be reversed during short recording from -20ms to +23ms for accelerometer vs gyroscope in Pixel6a (Fig.5). How big of the effect it can make on the analyses needs to be assessed before doing any calculations. Moreover, from our observations, obtained at 50Hz sampling rate, combined delays may become comparable to 20ms, the interval between the points, for some sensors. Slightly lower sampling would help with this bias. As it can be seen from available open sources, 30-40Hz acquisition frequency may produce quite stable data sets, though in foreground short recording sessions [8]. Predictability rates vs sampling frequency and amount of storage needed for it can be estimated from published work [11][12][13] and it also points at the most optimal values between 20-50Hz.

## V. Conclusion

Operating system delays are inherent unwanted behavior of the mobile devices when continuously accessing sensor data. Different manufacturer and device's movement may exacerbate this behavior. These delays may be a determining factor when collecting at high sampling rates that has to be evaluated beforehand, so the proper rate could be used to insure no loss of data occurs. In our experiment less than 40Hz sampling rate was found to be the highest limit that one can use on Google Pixel 6s and 16Hz on Samsung S22 for lossless data acquisition from all sensors.

## References


[1] Aguileta A.A., Brena R.F., Mayora O., Molino-Minero-Re E., Trejo L.A. (2019). Multi-Sensor Fusion for Activity Recognition—A Survey. Sensors (Basel). 19(17): 3808.

[2] Stisen A., Blunck H., Bhattacharya S., Prentow T.S., Kjærgaard M.B., A. Dey, Sonne T., Jensen M.M. (2015). Smart Devices are Different: Assessing and MitigatingMobile Sensing Heterogeneities for Activity Recognition. SenSys '15: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems. pp. 127–140.

[3] Grewal, M., & Andrews, A. (2010). How good is your gyro [ask the experts]. IEEE Control Systems, 30, 12–86.

[4] Weinberg, H. (2011). Gyro Mechanical Performance: The Most Important Parameter. Retrieved from http://www.mouser.cn/pdfdocs/ADI_MS2158_TechnicalArticle.PDF

[5] Straczkiewicz M., Keating N. L., Thompson E., Matulonis U.A., Campos S.M., Wright A.A., Onnela J. (2023). Validation of an open-source smartphone step counting algorithm in clinical and non-clinical settings. medRxiv. doi: 10.1101/2023.03.28.23287844

[6] Wang, J.-S., Hsu, Y.-L., and Liu, J.-N. (2010). An inertial measurement-unit-based pen with a trajectory reconstruction algorithm and its applications. IEEE Transactions on Industrial Electronics, 57(10):3508-21.

[7] Toyozumi, N., Takahashi, J., and Lopez, G. (2016). Trajectory reconstruction algorithm based on sensor fusion between imu and strain gauge for stand-alone digital pen. In Roboti

[8] Vaizman Y., Ellis K., Lanckriet G. (2017). Recognizing Detailed Human Context In-the-Wild from Smartphones and Smartwatches. IEEE Pervasive Computing, vol. 16, no. 4, October-December 2017, pp. 62-74.

[9] Blunck, H., Bouvin, N. O., Franke, T., Grønbæk, K., Kjaergaard, M. B., Lukowicz, P., & Wüstenberg, M. (2013). On heterogeneity in mobile sensing applications aiming at representative data collection. Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, 1087-1098.

[10] Reips U.-D., Buchanan T., Krantz J. H., & McGraw K. (2015). Methodological challenges in the use of the Internet for scientific research: Ten solutions and recommendations. Studia Psychologica, 15, 139–148.

[11] K.C. Liu, C.Y. Hsieh, S.P. Hsu, C.T. Chan. (2018). Impact of Sampling Rate on Wearable-Based Fall Detection Systems Based on Machine Learning Models. IEEE Sens. J., 18 (23), pp. 9882-9890

[12] P. Bet, P.C. Castro, M.A. Ponti. (2019). Fall detection and fall risk assessment in older person using wearable sensors: A systematic review. Int. J. Med. Inform., 130, Article 103946

[13] N. Zurbuchen, A. Wilde, P. Bruegger. (2021). A Machine Learning Multi-Class Approach for Fall Detection Systems Based on Wearable Sensors with a Study on Sampling Rates Selection. Sensors, 21 (2021), p. 938