**Abstract**

Sensor data reliability plays a critical role in applications that collect and uses this data. The types of effects that can invalidate the data could be physical data drift, bias, missing data or outliers, to name a few. These can arise from a number of factors, power management designed and operating system, that are built to conserve the power, are of the most concerns. This study assessed an impact of the battery charge and functional state of the device on the variability in the timestamps of the sensor data readings, where variability reflects the consistency of the sensor data readings. This study has found that lower battery charge alone had a tendency to or significantly lowered variability. High CPU loads significantly impacted (increased or decreased, depending on the condition) variability. The number of sensors similarly affected the variability, and also made it negligible or even reversed cross-correlation between them. These findings indicate that the process of the sensor data reading is highly dependent on the device’s conditions that may, as a result, impact the reliability of the data.

**Aim of the project:** Estimate the impact of available battery power in smart devices on variability of timestamps of data captured from device’s sensors.

**Research design:** Sensors are always or periodically active. They produce data at certain acquisition frequency in real time that are digitized and put it in a memory buffer for reading [1]. However, in cases where the sensors are managed by the Android OS, the sensor raw data acquisition can be delayed causing forward shifts in the timestamps. That delay can be random and not constant but rather a variable that depends on many parameters [2]. In this study we are concentrating on how device’s power management affects sensing data quality. While we can control the sensor data read frequency, we do not have access to the wait in a queue for the buffer read. Therefore, we do not have access to the time when sensor updates the buffer. However, sensing is much faster (>100Hz) [3] than we will read the sensor data (5Hz), hence, should not introduce significant differences in the timestamps by itself. In addition, and for the future research, we do have access to the Qualcomm sensing information through the QSEE API. For the QSEE supported sensors, we can compare OS (Android) reported sensing with the QSEE hardware timestamps for all supported sensors (3-4).

The timestamp variability will be used to estimate reliability of the sensors data timestamps under different conditions to understand better sensing data quality. Data quality can be assessed by evaluating the time interval between captured sensor data. We will use in-house designed mobile app to listen for specific sensor(s) at a given polling rate. App will be kept in the background or foreground during recordings. Since, the time when data are read from the sensor buffer are not going to be exactly equal to preset polling rate but dependent on OS queuing it, that ultimately will introduce a variability between the readings. We plan to quantify the sensing data quality for single and multiple sensors under normal conditions and when the device exhibits low power and high CPU utilization.

Initially, we select to monitor accelerometer and gyroscope sensors, which are most commonly used for health and other research [4, 5, 6, 7].

**Methods**

For this study we chose Google Pixel 6a smart phone with Android 11 operating system. It is modern and powerful device with variety of sensor data that it can capture. It has advanced power management system and should be ideal for the experiment we are planning to conduct.

***Single sensor study.***

*Our control* experiment for sensor data collection will be comprised of the collecting data timestamps on each sensor separately on 100% charged plugged-in device. This is best case scenario that we should expect. So, variability in the timestamps from the sensors data is expected to be at its minimum.

*The first condition* is 100% charged not plugged-in device, this condition will provide preferable real-world conditions in the course of relatively consistent data collection. This should reveal timestamp variability that one should expect during a daily use of their device that maybe slightly greater than in control.

*The second condition* is 10% battery charge remaining not plugged-in to imitate data recording under the various power conserving mechanisms utilized by the device’s hardware and OS. This is expected to produce the greatest variabilities in the readings that will provide insight into power management control of and access to the sensor’s data. For this scenario, we will consider a device that is in normal battery power mode and low power mode.

*The third condition*, 100% charged device under the heavy nearly 100% computational load to reveal how device’s power management systems enforce task threads depending on their power consumption. In this experiment our app tasks power consumption should be negligible to the load we are going to artificially introduce. The results will show what kind of competition profile there is implemented and how users’ activity may impact accuracy of the data coming from sensors.

***Multi-sensor study***

*In the next set of experiments*, we will simultaneously record data from both sensors using the same recording paradigm. The variability of timestamps between the two sensors will be assessed. This data will elucidate the impact on reliability of reading from multiple sensors in various conditions. In addition to the between sensors variability we can assess their cross-correlation (lag). If in fact it is pronounced and increased or decreased that would indicate that there is a relationship that power management introduces to multi-sensor data readings.

**Data and Statistical analyses**

Data on variability as means and standard deviations can be compiled from our experiments and statistically analyzed for significant differences. The variability of the mean interval between sensor readings across different collection conditions will be analyzed for the significant changes using Excel 2 tailed t-test function. In the situations where t-test did not show significant changes, the distributions of the intervals between sensor readings will be analyzed using two sample Z-score test formular written in Excel. 3D sensor data converted into a simple geometrical distance or rather acceleration absolute value would be routinely tested against intervals between sensor recordings 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. Throughout the text of this manuscript averages will be expressed as a mean ± standard deviation (SD) followed by number of measurements.

**Results**

Overall, variability of the interval between sensor data readings showed little differences in means across all the conditions. However, there were instances where it was profound, specifically when the device was plugged-in or asleep. This may refer to the proprietary manufacturer’s algorithm or the design of the app that is accessing sensor information. CCF analyses did not show correlation between the intervals and sensor reading. Normally, R was less than 0.1 (data not shown), meaning that sensor function itself does not, as expected, impose significant stress on the device functionality [1].

Most of the documented differences were based on the distribution analyses, indicating that there were notable non-homogeneous populations of the intervals, i.e. the times between the instances when app was allowed to capture data from sensors. In some examples there were extra peaks in distributions and in some distributions were skewed.

**Accelerometer** study has revealed that when Google Pixel 6a device was plugged-in, for both when app was in background and foreground, the interval between instances the app accessed accelerometer sensor’s data was very close to the polling rate of 200ms, about 201.86±4.89 (n=891) (Fig.1, A and B). At closer inspection of the distributions (Fig.2), it was discovered that there were actually two populations with averages 200 and 218ms. Causes of this at this time are unknown and maybe investigated in the different study.

Between the rest of the cases there was no significant differences in means, so interval distributions were analyzed with Z-test. This test revealed that there was significant effect of the high CPU load (Fig.1, G, 291.44±0.56, n=617) on the intervals when compared with just 100% charged device (Fig.1, C and D, 291.37±0.67, n=617 and 291.37±0.71, n=617). 10% charged devices (Fig.1, E and F, 291.42±0.76, n=617 and 291.39±0.77, n=617) showed tendency but were insignificant with Z scores of 0.6 and 1.3, respectively. High CPU load significant effect seems to be due to the lower standard deviation in the intervals distribution that indicates more consistent sensor readings under these conditions. Surprisingly, low battery charge conditions, though had higher SD, indicating wider distribution, did not result in significant difference from neither fully charged nor high CPU load situations.

Figure 1. Plugged-in state and high CPU load has a significant impact on the intervals between accelerometer sensor’s data access.

Chart, bar chart

Description automatically generated

\*, # show significant mean differences in intervals between sensor readings (t-test, p<0.01) for 100%, 10% and 100% charged with high CPU load when compared with (A, B) the plugged-in, 100% charged device with sensor reading app running in foreground and background, respectively. + depicts significant difference between (G) high CPU load and 100% charged device distributions of intervals between sensor readings (Z test, score >1.645, alpha 0.05). A - 100% charged device, plugged-in, app is in foreground, and B when app is in background. C -100% charged device, app is in foreground, and D when app is in background. E - 10% charged device, app is in foreground, and F when app is in background. G - 100% charged device, CPU is in high load, app is in background.

Figure 2. Accelerometer sensor reading intervals show two modes when device is plugged-in.

Graphical user interface, application

Description automatically generated

**Gyroscope** study, unlike accelerometer one, showed no significant differences in the intervals between the instances when gyroscope data was recorded across all the cases. Analysis of distributions with Z test revealed significant differences again between high CPU load intervals 291.45±0.57 (n=617), Fig.3, G. Namely, with 100% charged device both when data collecting app was in foreground and background 291.38±0.67(n=617) and 291.38±0.77(n=617) and with 10% charged device when the app was in background 291.38±0.79(n=617) (Fig.3 C,D, and F). Plugged-in device in both data collecting scenarios and 10% charged device when data collecting app was in foreground (Fig. 3, A, B, E) showed high but insignificant tendency (Z scores were 1.4, 1.5, and 1.5, respectively). Similar to the accelerometer study, SD was the lowest in high CPU load case and seemed to cause the significant Z score. Gyroscope at low battery charge showed significant effect (due to greater SD, i.e. variability in the intervals) when compared with high CPU load, in contrast to accelerometer that only had a tendency, but only when app was in background.

Figure 3. High CPU load has a significant impact on the intervals between gyroscope sensor’s data access.

Chart, bar chart

Description automatically generated

+ depicts significant difference between high CPU load (G) and 10% and 100% charged device distributions of intervals between sensor readings (Z test, score >1.645, alpha 0.05). A - 100% charged device, plugged-in, app is in foreground, and B when app is in background. C -100% charged device, app is in foreground, and D when app is in background. E - 10% charged device, app is in foreground, and F when app is in background. G - 100% charged device, CPU is in high load, app is in background.

**Accelerometer and Gyroscope** sensors data recording were done simultaneously in this set of experiments. It has been observed that similarly to Accelerometer data, obtained during single sensor recordings, when device was plugged-in, for both when data collecting app was in background and foreground, the interval between the readings was very close to the polling rate of 200ms, 202.08±5.13 (n=890) and 202.18±5.28 (n=890), respectively (Fig.4, A1 and B1). Interval distributions (Fig.5) were found to be similar to Fig.2 and had two modes. Averages for accelerometer sensor intervals in the plugged-in device were significantly lower from the rest of the accelerometer data sets (Fig.4).

Multi-sensor readings when compared to single-sensor ones did not show significant changes when distributions of the intervals measured in the device that was not plugged-in were tested with Z-test. There were tendencies, though insignificant, in Gyroscope readings between high CPU load (Fig.4, G2) with Z scores of around 1.0 for Fig. 4, A2, B2, C2, D2, and F2. Moreover, SD in high CPU load showed higher values of 2.34 and 4.2 then the rest of the measurements that had SD around 0.7. This was a complete reversal from the single-sensor experiments described above and only at a specific condition. That is, multi-sensor reading appears to introduce greater impact on the consistency of the intervals between the readings at high CPU loads, hence it may possibly reduce the quality of the sensor data, under these circumstances. Just to note, single-sensor reading showed reduction in SDs at high CPU loads. Other conditions, plugged-in, 10% and 100% charged phones showed relatively similar SDs regardless of the number of sensors monitored.

Figure 4. Plugged-in state has a significant impact on the intervals between accelerometer sensor’s data access.

Chart, bar chart

Description automatically generated

\*, # show significant mean differences in intervals between accelerometer sensor readings (t-test, p<0.01) for 100%, 10% and 100% charged with high CPU load when compared with the plugged-in, 100% charged device with sensor reading app running in foreground and background, respectively. Indices 1 and 2 refer to Accelerometer and Gyroscope interval measurements, respectively. A - 100% charged device, plugged-in, app is in foreground, and B when app is in background. C -100% charged device, app is in foreground, and D when app is in background. E - 10% charged device, app is in foreground, and F when app is in background. G - 100% charged device, CPU is in high load, app is in background.

Figure 5. Accelerometer sensor reading intervals show two modes when device is plugged-in, multi-sensor reading.

Graphical user interface, application

Description automatically generated with medium confidence

**Device movement**

As we mentioned above that when device was asleep, we saw the greatest increases in the intervals between the polling. Assumption was made that operating system due to device and sensors inactivity was disregarding application requests to access sensor’s data. To investigate if these numbers can be affected by the device movements, we conducted a separate set of measurements with the devices mounted on a robotic arm from Elephant Robotics and constantly moved about with frequent changes in directions to make sure accelerometer and gyroscope sensors get engaged. The same trajectory of the movement, stored in the robotic arm memory, was used for all the data collection in these series.

**Accelerometer** data from stationary or moving Google Pixel 6a phone are presented on Figure 6. 1st bar (Fig.6A) shows data from plugged-in to a charger device just for the illustration purposes. We did not use plugged-in device in our moving experiment due to the constrains that tethering imposes on a setup that might serve as a confounding factor. Nevertheless, distributions of between polling intervals of stationary plugged-in (Fig.6A), 100% (Fig.6B), and 10% (Fig.6D) charged devices (1270.23±5996.30 n=117, 1498.39±5285.21 n=84, 2210.84±5257.84 n=77, respectively) were significantly different from 100% high load data (Fig.6F; 291.44±0.54 n=617) with Z scores of 1.76, 2.09, 3.2, respectively. Similarly, moving 100% (Fig.6C), and 10% (Fig.6E) charged devices (2191.94±9134.04 n=82, 1593.06±4258.59 n=112, respectively) were different from 100% high load (Fig.6G; 291.48±0.54 n=617) with Z scores of 1.88 and 3.24, respectively. These results highlight remarkable differences between sleeping and, though under a high amount of stress but still awake devices. Whether stationary or moving, judging by the number of observations, accelerometer data are much rarely accessed in a sleeping state.

Sleeping stationary vs. moving comparison although insignificant however still showed a tendency. Namely, 100% charged when moving showed on average longer between polling interval (Fig.6B vs. C, Z score=0.6). 10% charged phones when moving revealed the opposite, i.e. on average shorter polling intervals (Fig.6D vs. E, Z score=0.86). In the high load case, intervals were virtually identical (Fig.6F vs. G, Z score=1.2).

Figure 6. Moving and stationary devices showed similar mean intervals between the readings of accelerometer data with distributions significantly different from high CPU loads.

Chart

Description automatically generated

+ shows significant difference in the distributions of intervals between sensor readings compared with (F) high CPU load when device was stationary (Z test, score >1.645, alpha 0.05). ^ shows comparisons with (G) high CPU load when device was moving. A - 100% charged plugged-in, B – 100% charged, D - 10% charged device sleeping and stationary. C – 100% charged, E – 10% charged, device sleeping and moving. F and G – high CPU load, with app in background, when device was stationary or moving, respectively.

Tendencies noticed in the Figure 6 were investigated further. Figure 7 shows whisker plots of the intervals between the pollings for 100% charged and 10% charged devices. High CPU load cases did not show visible differences and were not shown. Surprisingly, device movement while it was asleep produced clusters of even longer intervals between the accelerometer data readings with averages around 40s and 15s (Fig.7Bs) when compared with stationary phones intervals of 20s and 5s (Fig.7As), for 100% and 10% charged devices, respectively. Despite that moving phones averages were close or lower than stationary (Fig.6), indicates that device was actually able to capture accelerometer data more often at the faster rate, approximately every 291ms.

Figure 7. Accelerometer data of moving devices shows clusters of longer intervals between sensor data reading.

Chart, scatter chart

Description automatically generated Chart, scatter chart

Description automatically generated

A – denotes whisker box plot for sleeping stationary device, when B – sleeping moving device.

**Gyroscope** data from stationary or moving Google Pixel 6a phone are presented on Figure 8. 1st bar (Fig.8A) shows data from plugged-in to a charger device just for the illustration purposes. The tests showed that only the distribution of between polling intervals of stationary plugged-in (Fig.8A, 1534.11±5221.82 n=84) was significantly different from high CPU load data (Fig.8F; 291.45±0.57 n=617) with Z scores of 2.18. 100% (Fig.8B), and 10% (Fig.8D) charged devices (11410.33±33398.14 n=91, 6237.89±15117.17 n=9, respectively) were not significantly different from 100% high load data (Fig.8F) though showing a tendency with Z scores of 1.00, and 1.18, respectively. However, moving 100% (Fig.8C), and 10% (Fig.8E) charged devices (821.74±4344.30 n=186, 1246.55±3223.11 n=131, respectively) were different from 100% high load (Fig.8G; 291.46±0.56 n=617) with Z scores of 1.67 and 3.39, respectively. These results show that different sensors may react differently when device is in a sleeping mode depending on a battery charge remaining. It appears that gyroscope data are more often accessed at lower battery charge when device is in a sleeping state and in motion.

Sleeping stationary vs. moving gyroscope data comparison, in contrast to accelerometer data, gyroscope’s between readings interval distributions showed significant differences between stationary and moving devices when they were 100% charged and sleeping (Fig.8 B vs. C, Z score = 3.01). 10% charged devices data distributions were insignificant but showed a tendency (Fig.8D vs. E, Z score=0.99). Both cases when moving revealed that on average there were shorter polling intervals between the readings. In the high CPU load case, intervals were virtually identical (Fig.8F vs. G, Z score=0.3).

Figure 8. 100% charged sleeping but moving device reads gyroscope data more often.

Chart, box and whisker chart

Description automatically generated

+ shows significant difference in the distributions of intervals between sensor readings compared with (F) high CPU load when device was stationary (Z test, score >1.645, alpha 0.05). ^ shows above comparisons but with (G) high CPU load when device was moving. & shows significant difference in distribution of between sensor reading intervals between stationary and moving devices under the same conditions (100%, 10% charged, and high CPU load). A - 100% charged plugged-in, B – 100% charged, D - 10% charged device sleeping and stationary. C – 100% charged, E – 10% charged, device sleeping and moving. F and G – high CPU load, with app in background, when device was stationary or moving, respectively.

Closer look at the distributions of the intervals between gyroscope sensor data reading (Fig.9) for 100% and 10% charged devices revealed opposite relationships when compared to accelerometer (Fig.7). Figure 9 shows whisker plots of the intervals between the readings for 100% charged and 10% charged devices. Device movement while it was asleep (Fig.Bs) produced more compact clusters of shorter intervals between the gyroscope data readings with averages around 10s (Fig.9Bs) when compared with stationary phones intervals of 50s and 20s (Fig.9As), for 100% and 10% charged devices, respectively. Ultimately, that made moving phones averages lower than stationary, that was similar to the effect observed in accelerometer data, however displaying different or opposing dynamics.

Figure 9. Gyroscope data of moving devices shows clusters of shorter intervals between sensor data reading.

A screenshot of a graph

Description automatically generated with medium confidence A picture containing text, screenshot, number, font

Description automatically generated

A – denotes whisker box plot for sleeping stationary device, when B – sleeping moving device.

**Accelerometer and Gyroscope** sensors data recording while device was moving, or stationary were done simultaneously in this set of experiments. Averages and SDs of the between reading intervals are presented on Figure 10. Figure 10 A1 and A2 illustrate tethered device data while stationary and asleep, similar to Figures 6 and 8. Analyses did not reveal significant changes in averages or distributions between the moving and stationary devices. However, like accelerometer alone data, accelerometer and gyroscope data showed significant changes in distributions of intervals between reading of plugged-in, 100% and 10% charged devices when compared with devices at high CPU load (Accelerometer: Fig.10A1,B1,D1 vs F1, 2284.50±6314.87 n=69, 1841.26±7085.69 n=214 with Z score=2.03, 1583.70±5122.55 n=220 with Z score=2.22, vs 291.37±2.38 n=617; Gyroscope: Fig.10A2,B2,D2 vs F2, 3093.29±7200.04 n=51, 1862.41±7147.52 n=217 with Z score=2.03, 1527.34±2898.29n=84 with Z score=2.21 vs 291.19±4.18 n=618). Analogous results were seen in the distributions when device was moving (Accelerometer: Fig.10C1,E1 vs G1, 2495.22±8213.45 n=54 with Z score=1.97, 1248.40±2898.29 n=144 with Z score=3.96 vs 291.54±3.42 n=617, Gyroscope: Fig.C2,E2 vs G2, 2496.83±8217.36 n=54 with Z score=1.97, 1231.32±2877.65 n=146 with Z score=3.95).

Figure 10. Moving and stationary devices showed similar mean intervals between the readings of simultaneously acquired accelerometer and gyroscope data.

Chart, bar chart

Description automatically generated

+ shows significant difference in the distributions of intervals between sensor readings compared with (F) high CPU load when device was stationary (Z test, score >1.645, alpha 0.05). ^ shows above comparisons but with (G) high CPU load when device was moving. Indices 1 and 2 refer to Accelerometer and Gyroscope interval measurements, respectively. A – depicts 100% charged plugged-in device data. For 100% charged device B - shows stationary, and C – moving device data. For 10% charged device D – shows stationary, and E – moving device data. For 100% charged device at high CPU load F - shows stationary, G – moving device data.

For the consistency with previous data presentation, whisker box plots were used for simultaneously recorded data from two sensors too (Fig.11), even though there was not significant differences and also reliable tendencies. Since we recorded data simultaneously, we had an opportunity to check recordings for cross-correlation and see if there is a similarity between two sensors timing. Cross-correlation functions were calculated using intervals between sensor readings as accelerometer vs gyroscope for each experimental condition (Fig.12). 100% and 10% charged devices revealed robust correlation between the sensors around zero lag ±1ms and with some longer positive and negative lags. The most interesting result was observed when devices were at the high CPU load. In this case when device was stationary there was a little cross-correlation between the sensors, and when device was moving, it actually showed great double peak (with negative lags of 1 and 5ms) negative correlation. These observations can be supported by reviewing the outliers on Figure 11. It can be noticed that there is a similarity in the data points between accelerometer and gyroscope in 100% and 10% charged devices.

Figure 11. Simultaneously obtained Accelerometer and Gyroscope data of stationary and moving devices.

A screen shot of a graph

Description automatically generated with medium confidence A screen shot of a graph

Description automatically generated with medium confidence A picture containing text, screenshot, number, font

Description automatically generated A screen shot of a graph

Description automatically generated with medium confidence

A – denotes whisker box plot for sleeping stationary device, when B – sleeping moving device.

Figure 12. Functional state (high CPU load) of the device showed greatest desynchronizing impact on between sensor cross-correlation.

A picture containing text, screenshot, line, number

Description automatically generated Chart, line chart

Description automatically generated A picture containing text, number, screenshot, font

Description automatically generated Chart

Description automatically generated Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Closer inspection of the actual intervals of both accelerometer and gyroscope readings at 100%, 10% charged devices and Hi CPU load is shown at Figure 13 for both stationary and moving conditions, in the same order as on Figure 12. Sensor data was plotted out here as a raster plots. Namely, for each reading at the time sensor was read we plotted arbitrary 1, otherwise we plotted a zero. Only about 1 or 2s of data points were displayed on these graphs for the clear view. All raster plots showed more or less persistent patter of readings. However, Hi CPU load, stationary case revealed changes in the timing of the reading. It was almost synchronous in the beginning (see inset) and then it gradually separated, the gyroscope readings were progressively delayed in course of the recording. That, ultimately resulted the least cross-correlated picture on Figure 12. The only negative cross-correlation in Hi CPU load, moving case can be attributed to the fact that timing of the accelerometer sensor reading was in the opposite relationship with gyroscope, i.e. if accelerometer is read often, then gyroscope is read with a delay. It was hard to notice on the Figure 13, cause it happened on a millisecond scale, but it must have been very consistent and resulted in the negative correlations.

Taken together it appears to show that accessing multi-sensor data could be affected in positive or negative way by itself and by the functional state of the device. That may impact data reliability in unwanted way.

Figure 13. Sample raster plots of Accelerometer and Gyroscope reading times.

A graph with blue and orange lines

Description automatically generated with low confidence A picture containing text, screenshot, line, plot

Description automatically generated A picture containing text, screenshot, line, plot

Description automatically generated

A picture containing text, screenshot, line, plot

Description automatically generated A graph with blue and orange lines

Description automatically generated with low confidence A picture containing text, screenshot, line, plot

Description automatically generated

Top line represents instances when accelerometer and the bottom one when gyroscope data were read. Left axes displays arbitrary units for accelerometer data, right axis for gyroscope data. 1 represents reading and 0 waiting state. Both axes scales were extended to be able to display for plots on the same panel in Excel. Bottom axis shows the time of recording in ms divided by 2. Plots show data recorded approximately in the middle of the experiment. Inset in “Hi GPU load, stationary” plot shows the data recorded in the beginning of the data collection procedure.

**Discussion**

**Stationary awake devices**

Sensor data reliability is very important in various scenarios of smartphone usage from user perspectives and especially for research requirements. Utilization of the sensors data is still limited and resorted to specific and easily identifiable phenomena, step, fall, jump etc [8, 9, 10]. In general approach to the data easily identifiable activity may not be present, thus, it is imperative for the data to be reliable and usable, especially when trying to develop learning algorithms on the sensors data one must be sure in their accuracy and consistency. Acquisition frequency is also a great factor in usability of the sensors data, the general rule is the higher frequency the better [11]. This will introduce higher battery consumption and may discourage end users from it. Moreover, it has been reported that there is a great deal of drift in the data that must be cleaned out by using Fourier transform and band pass filtering mostly to attenuate low frequency noise, as described for example here [12]. This indicates an existence of signals with longer durations that affect the sensor readings.

Similarly, in our results, although we saw that overall there was no significant changes in average times between sensor readings, but there was noticeable change in distributions with longer times between the readings. Especially, it was pronounced when devices were sleeping. So, the most reliable conditions for sensor data acquisition would be when devices are active. However, even in this state there was about 90ms delay from the polling rate. This was most likely introduced by the operating system itself and would be hard to overcome in normal settings. Surprisingly, lower battery charge did not introduce significant deviations in the intervals between the readings. High CPU load was also similar with the rest of the averages. Taken together, it may indicate that our device, that was the latest model of Android phones from Google, was able to handle the tasks that we programmed them to do without much of the effort. So, accessing the sensors data by itself is low power procedure.

**Sleeping and moving devices**

When mobile device enters a sleep mode all nonessential threads get suspended. Thus, our data acquisition was basically quickly stopped, however triggered from time to time. One fact should be mentioned here is that lower battery charge showed beneficial impact on the intervals between sensors reading. Namely, sensors were read 1.5-2 times more often. This trend even improved when we introduced movement to the sleeping devices. One explanation can be because data were accessed from the cache and not actually from the memory buffer that is managed by PMI and OS or it was something similar to the jitter effect in smartphones [13, 14]. The fact that lower battery charge has benefits will require further exploration to understand what drives this behavior.

**Multi-sensor reading**

Overall whether we accessed single or multiple sensors at the same time, the averages and distributions of intervals between the readings were similar. That allows us to conclude that all sensors work separately and accessing their data is an independent process. Cross correlation analyses between simultaneously read sensors without high CPU load revealed high positive correlation with the major lag in the middle (around few ms) and minor lags at 5ms, 17ms. It may mean that greater intervals between the readings of one sensor are followed by greater intervals in the other. That fits the assumption that the system is controlling the access to all sensor data and that makes them work in unison.

However, high CPU loads revealed no, when stationary, or strong negative correlation with the lag peak value -1ms, when device was moving. Thus, indicating that greater intervals between the reading one sensor led to shorter ones in the other. Moreover, intervals were continuously changing between Accelerometer and Gyroscope when device was stationary. From the analyses of the distributions, it was also noted that high CPU loads had much greater variability that was the opposite of the single sensor case. That increase in the variability is to be expected when the number of sensors involved in the recordings increases. Perhaps high CPU load imposed enough strain on the system itself or on the processing power that leads to this effect when one sensor is favored when compared to the other. That potentially may introduce a weakness in the data set of one of the sensors.

**Battery charge and functional state**

We investigated 100% and 10% battery charge effect on distributions and averages of the intervals between sensor data readings. Interval averages in both conditions were close to each other. Also, 100% charged devices distributions were significantly different and 10% charged mostly had only tendency to be different when compared to Hi CPU load condition. However, data from 10% charged devices was not different from 100% ones.

Moreover, and unexpectantly, lower charged device showed better consistency in accessing sensor data. The improvements that we saw were most likely due to specific algorithms that OS utilizes in low battery mode, or artifacts [13, 14], that was why they were documented as an improvement. This might have lowered reliability of the data collected from sensors under these conditions too.

Thus, we cannot directly deduce impact of the low charge on sensor data reliability/validity without further study. However, we can say that high CPU load had significant impact on the variability and correlation in sensor data reading. Distributions and cross-correlations under this condition showed the most drastic changes that may indicate poor data quality.

**Conclusion**

Low battery charge and functional state of the device may introduce unwanted data inconsistencies when capturing sensor information. Especially, this data may be vulnerable when accessing several sensors at the same time. Although, question remains whether inconsistently recorded data are valid and still usable. Certain data validation procedures have to be put in place or special acquisition algorithms in app design must be devised before data can be deemed clean and usable for other research.

**References**

1. N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, “A survey of mobile phone sensing,” *IEEE Communications magazine*, vol. 48, no. 9, pp. 140–150, 2010.
2. Stisen A., Blunck H., Bhattacharya S., Prentow T.S., Kjærgaard M.B., Dey A., Jensen M.M. Smart devices are different: Assessing and mitigating gmobile sensing heterogeneities for activity recognition; Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems; Seoul, Republic of Korea. 23 November 2015; pp. 127–140.
3. G. Grouios, E. Ziagkas, A. Loukovitis, K. Chatzinikolaou, E. Koidou. Accelerometers in Our Pocket: Does Smartphone Accelerometer Technology Provide Accurate Data? Sensors (Basel). 2023 Jan; 23(1): 192.
4. J.R. Kwapisz, G.M. Weiss, and S.A. Moore. Activity recognition using cell phone accelerometers. SIGKDD Explorations Newsletter, 12(2):74–82, 2011.
5. T. Brezmes, J.L. Gorricho, and J. Cotrina. Activity recognition from accelerometer data on a mobile phone. Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, pages 796–799, 2009.
6. W. Wu, S. Dasgupta, E.E. Ramirez, C. Peterson, and G.J. Norman. Classification accuracies of physical activities using smartphone motion sensors. Journal of Medical Internet Research, 14(5), 2012.
7. Douangphachanh V., Oneyama H. Exploring the use of smartphone accelerometer and gyroscope to study on the estimation of road surface roughness conditions; Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics; Vienna, Austria. 1–3 September 2014; pp. 783–787.
8. Zhai Y., Nasseri N., Pöttgen J., Gezhelbash E., Heesen C., Stellmann J. Smartphone accelerometry: A smart and reliable measurement of real-life physical activity in multiple sclerosis and healthy individuals. Front. Neurol. 2020;11:688.
9. Culhan K., O’Connor M., Lyons D., Lyons G. Accelerometers in rehabilitation medicine for older adults. Age Ageing. 2005;34:556–560.
10. Chinrungrueng J., Sartsatit S., Intarapanich A. My act: An automatic detection of daily physical activity and calorie expenditure using smart phones. J. Assist. Rehabil. Ther. Technol. 2014;2:23187.
11. Yurur, O., Labrador, M. & Moreno, W. Adaptive and energy efficient context representation framework in mobile sensing. IEEE Trans. Mob. Comput. 2014; 13, 1681–1693.
12. Straczkiewicz, M., Huang, E.J. & Onnela, JP. A “one-size-fits-most” walking recognition method for smartphones, smartwatches, and wearable accelerometers. npj Digit. Med. 2023; 6, 29.
13. E. Peguero, M. Labrador and B. Cook, Assessing Jitter in Sensor Time Series from Android Mobile Devices. 2016 IEEE International Conference on Smart Computing (SMARTCOMP), St. Louis, MO, USA, 2016, pp. 1-8.
14. W. Ramadan, E. Ozer. Modal analysis under jittering and kernel clock distribution: single-output identification. 2023. Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction.