Accurate Reaction Times on Smartphones: The Challenges of Developing a Mobile Psychomotor Vigilance Task

Mel Arthurs
mel.arthurs@insight-centre.org
Insight Centre for Data Analytics
Glasnevin, Dublin 9, Ireland

José Juan Dominguez Veiga
jose.dominguezveiga3@mail.dcu.ie
Insight Centre for Data Analytics
Glasnevin, Dublin 9, Ireland

Tomás Ward
tomas.ward@insight-centre.org
Insight Centre for Data Analytics
Glasnevin, Dublin 9, Ireland

ABSTRACT

The mobile psychomotor vigilance task (PVT) has been found to be a valid predictor of cognitive fatigue. However, absolute reaction time (RT) recorded by mobile PVT is inaccurate. This is concerning as participant RTs are used in the analysis of PVT results. This paper aims to characterise this problem and assess the margin of error across common iOS software frameworks. A novel Arduino test instrument was developed to simulate a user's reaction, providing a ground truth for the RT. We found in our experiments that there is between a 29.57% and 48.58% increase over the ground truth RT in the iOS implementations tested. These are significant overestimations that will affect the validity of the outcome metrics for any mobile PVT study participants.

CCS CONCEPTS

• **Human-centered computing** → **Smartphones**; • **Applied computing** → *Life and medical sciences.*

KEYWORDS

PVT, cognitive fatigue, reaction times, latency, smartphones

ACM Reference Format:
Mel Arthurs, José Juan Dominguez Veiga, and Tomás Ward. 2021. Accurate Reaction Times on Smartphones: The Challenges of Developing a Mobile Psychomotor Vigilance Task. In *2021 International Symposium on Wearable Computers (ISWC '21), September 21–26, 2021, Virtual, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3460421.3478818

1 INTRODUCTION

The psychomotor vigilance task (PVT) is a test commonly used to assess cognitive fatigue [10, 12]. The test measures participant reaction times (RT) over a number of trials. Resulting RTs are indicative of participant alertness, where RT increases as cognitive performance decreases [18]. The display presents a stimulus to the participant after a randomised interval [3]. Participants must detect the presence of the stimulus, recognise the stimulus and react to the stimulus. Each of these three phases contribute to the participant reaction time for an individual trial. Existing implementations use the preferred touch down gesture to respond to the stimulus [10], which is also used in our implementations.

ISWC '21, September 21–26, 2021, Virtual, USA

ACM ISBN 978-1-4503-8462-9/21/09.
https://doi.org/10.1145/3460421.3478818

Originally designed to be undertaken in a lab based setting [5], there have been a number of mobile implementations studied to create a more ecologically valid assessment [2, 4, 10]. Although mobile PVT has been found to be an accurate predictor of cognitive fatigue [16], smartphones introduce a significant margin of error when measuring the absolute RTs[1]. Modern smartphones are not real-time systems and do not comply with hard deadlines [14].

There are a number of potential sources of latency on smartphone devices [2]. This is shown in Section 3 and 4 where mobile PVT can overestimate by up to 121.43ms. The PVT-192 is the gold standard device with an error of $\pm 1ms$ [11] and is used in numerous studies [2, 13]. Therefore, absolute RTs from mobile devices are significantly more inaccurate compared to conventional PVT devices.

This is especially concerning when assessing the PVT outcome metrics in which accurate RTs are required. This includes the mean RT, the median RT, the standard deviation of the RT, the fastest 10% RT, the slowest 10% RT, number of lapses [6]. Therefore, researchers need to be aware of the quality of RTs produced by smartphone devices and the potential effects on any conclusions drawn from these results.

One method used to offset the less accurate timing is by increasing the trial count [17]. However, this comes with the disadvantage of increasing participant burden [1]. By aiming to produce RTs with a precision on a par with traditional hardware devices, mobile PVT may become a more unobtrusive method of measuring cognitive fatigue. This paper makes the following contributions:

- In Section 2, an accurate and accessible method of automated testing for touchscreen based mobile PVT implementations is described. This will allow researchers to provide a guarantee of the accuracy of the RTs that their implementation produces for a chosen device(s).
- Although mobile PVT implementations may be valid at evaluating fatigue, the absolute RTs measured by smartphone devices will have a significant margin of error due to sources of latency. We analyse the latency of three PVT algorithm iOS implementations in Section 3.
- Furthering on this, in Section 4 we analyse the impact of the CPU usage (one of the possible sources of latency) on the MetalKit implementation.

2 METHODS

Three implementations for the PVT were created for iOS (MetalKit, UIKit, and WebKit)[2]. MetalKit is a framework which provides near direct access to the GPU on iOS [7]. The UIKit framework allows

[1] Where RT is the reaction time measured in milliseconds (ms).
[2] https://github.com/arthursmel/pvt-demo

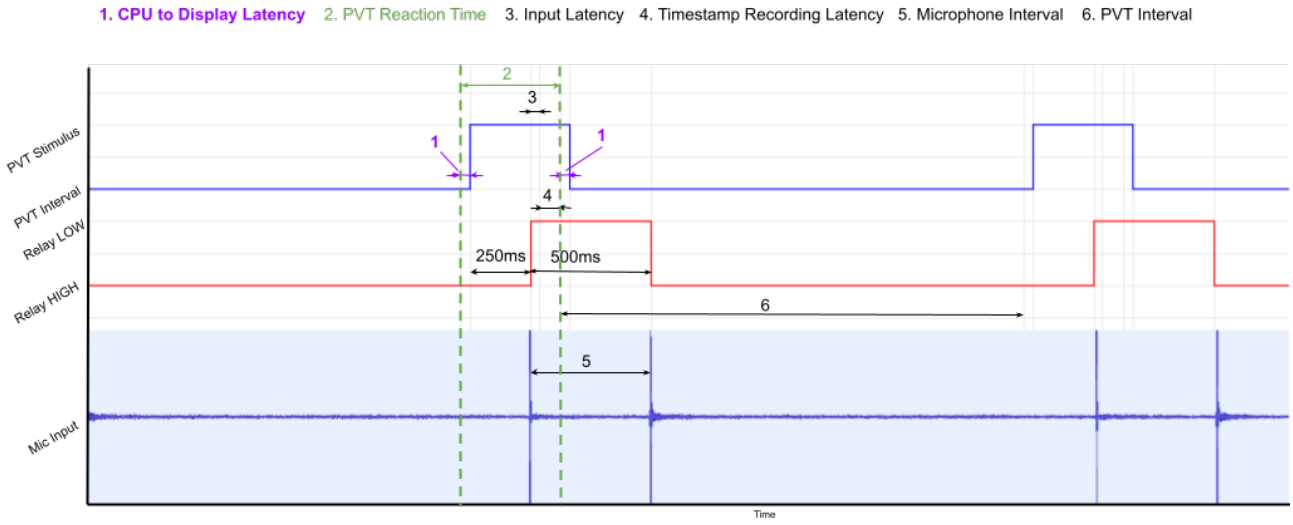

Figure 1: Two periods of microphone input, corresponding to two trials of the iOS-PVT

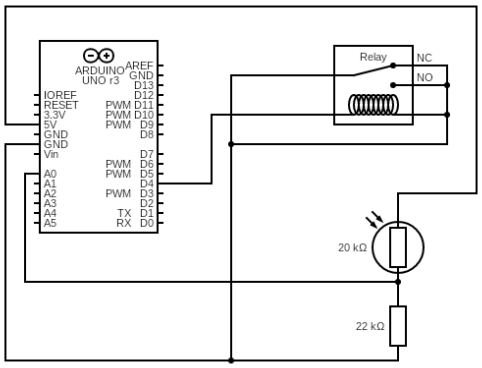

Figure 2: The circuit diagram of the Arduino test instrument

developers to create graphical event driven user interfaces on iOS. The WebKit framework offers a native solution to support Javascript on iOS. The following definitions are used in this paper:

(1) ***CPU to Display Latency*** is the time taken to execute a graphical update on the screen.
(2) ***PVT Reaction Time*** is the reaction time measured by the smartphone's PVT implementation.
(3) ***Input Latency*** is the time taken to register the user's touch on the screen and call the *touchesBegan* method.
(4) ***Timestamp Recording Latency*** is the latency caused by accessing the *Date* object to get the current unix timestamp and to update the PVT state.
(5) ***Microphone Interval*** is the time between the sound of the relay touching the screen and the sound of the relay releasing from the screen.
(6) ***PVT Interval*** is the duration between two trials. It is decided by a pseudorandom number generator with a range from 2000*ms* to 3000*ms*.

2.1 Measuring *CPU to Display Latency*

This metric was analysed in the Instruments profiling application on macOS[8]. The iPhone device was connected to the MacBook for profiling as described in Section 2.3.

2.2 Measuring *Input Latency*

The *Input Latency* caused by registering and handling touches using this method was measured using the ATI[3]. The idea behind this test is to send periodic touches from the Arduino (every 250*ms*) to the touch screen. The timestamps of when the touches were received are then recorded by the iOS device. Any variance from the period in the timestamps could then be attributed to the latency caused by the device in registering and handling the touches.

2.3 Devices

All experiments were performed on an iPhone 7 running iOS version 14.4.2. Wi-Fi and mobile data were switched off, Bluetooth disabled, low power mode disabled, and a battery level of 100%. Brightness was set to 100% and the device was charging via USB cable. The version of Instruments used to profile this application was 12.4 (12D4e). Profiling was only performed on the experiments when stated. The MacBook used to profile the application was a MacBook Pro 13 inch, early 2015 running macOS Big Sur v11.2.3 (20D91).

2.3.1 Arduino Test Instrument (ATI) Setup. The ATI exploits a non-latching relay's armature to simulate a finger touch. When a touch is to be simulated, 0V is applied to the copper tape on the screen by setting the relay's input to low for a short period of time (500*ms*). This distorts the screen's electrostatic field and registers a touch.

[3] Arduino Test Instrument.

Figure 3: The modified non-latching relay (side view & bottom view)

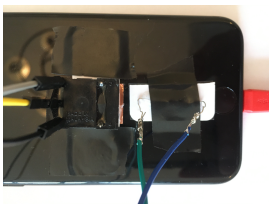

Figure 4

A square of copper tape (1.2*cm* x 1.2*cm*) was affixed to the touch-screen to simulate a user's fingertip. The state of the PVT must be known to decide when to apply a touch. The stimulus state and the interval state can be identified based on the brightness of the display. An LDR measures this screen brightness to decide if stimulus is being displayed - simulating the participant's eye. The LDR and relay circuit is described in Figure 2

An Arduino Uno R3[4] was used to implement the touch logic. An Arduino program simulates a touch when the screen brightness is over a certain value[5]. The program ensures only a single touch occurs when the screen brightness is over this value. By using a delay function in the program, a delay in the response of 250*ms* is applied to simulate an average human visual RT [9].

The LDR was secured to the iPhone screen using electrical tape as shown in Figure 4. The square of copper tape was affixed to the screen. The relay was secured on top of the copper tape square using electrical tape as shown in Figure 4.

2.3.2 Modified Non-Latching Relay. An Omron G5LE-1-DC5 SPDT non-latching relay was modified to simulate a finger touch. The outer casing was removed from the component to expose the armature. A plastic card (5.5*cm* x 1.8*cm*) was used as base to secure the component to the touch screen. This card was hot glued to the relay legs as shown in Figure 3. A strand of silver plated copper wire of length 2.5*cm* was bent into a hook shape as shown in Figure 3. It is important to note that this hook should not touch the screen when the input voltage is high and should be touching the copper tape when the input voltage is low. This wire was then soldered to the armature. The accuracy of the 500*ms* Arduino delay was verified using a lavalier microphone (model: X0011G1FBN). This is shown in Figure 1, where the amplitude peaks are caused by the relay touching the screen.

[4]https://store.arduino.cc/arduino-uno-rev3

[5]Value varies across devices; a value of 500 was used for the iPhone with the maximum screen brightness enabled in the device settings.

Figure 5: The LDR housing (top view & bottom view)

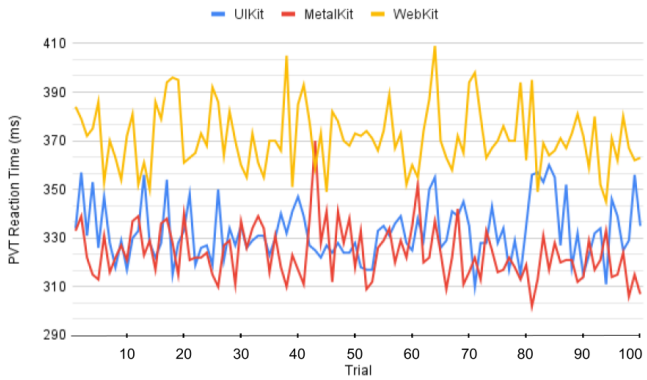

Figure 6: The PVT Reaction Time (ms) for 100 Trials of Each Implementation

2.3.3 LDR Housing. A through hole LDR (NSL-19M51) was housed to allow for the attachment to the device's screen. The LDR was hot glued to a piece of plastic card (3*cm* x 1.5*cm*). A clear plastic card with the same dimensions was hot glued to the underside of the previous card to protect the LDR as shown in Figure 5. A hole ⌀0.5*cm* was punched through the card as a frame for the LDR.

3 THE EFFECT OF SOFTWARE FRAMEWORKS ON PVT RESULTS

The software framework used to implement the PVT algorithm affects the participant RTs. This paper compares the RTs measured by the MetalKit, UIKit, and WebKit PVT implementations on iOS. A valid smartphone based PVT aims to minimize the margin of error to produce accurate RTs.

The PVT test instrument described in Section 2.3.1 was used to periodically simulate a reaction to the stimulus with a 250*ms* delay over 100 trials. The *PVT Reaction Time* was collected for each trial. This experiment was repeated with each implementation.

3.1 MetalKit Software Latency

It is expected that the MetalKit implementation would result in the most accurate *PVT Reaction Time* out of the implementations tested, with a 29.57% increase over the ground truth RT (250*ms*) as shown in Table 1. The Metal framework offers direct communication with the device GPU which reduces overhead in rendering on-screen graphics, providing a high level of performance [7].

[6]The sum of *CPU to Display Latency*, *Input Latency*, and *Timestamp Recording Latency*.

[7]*CPU to Display Latency* may only be profiled when using MetalKit graphical operations.

Framework	PVT Reaction Time	Implementation Latency[6]	CPU to Display Latency	Input Latency
MetalKit	323.93ms (SD=10.91ms, n=100)	73.93ms	19.00ms (SD=11.74, n=100)	0.34ms (SD=4.38ms, n=100)
UIKit	332.59ms (SD=11.83ms, n=100)	82.59ms	Unknown[7]	0.34ms (SD=4.38ms, n=100)
WebKit	371.43ms (SD=13.13ms, n=100)	121.43ms	Unknown	0.19ms (SD=5.05ms, n=100)

Table 1: Software Latencies of the PVT Algorithm Using Common iOS Software Frameworks with Arduino Delay of $250ms$

CPU Usage	PVT Reaction Time	Implem. Latency
Normal	324.96ms (SD = 12.21ms, n=100)	74.96ms
High	334.76ms (SD = 13.88ms, n=100)	84.76ms

Table 2: Comparison of the MetalKit *Implementation Latency* with 'normal' CPU usage and 'high' CPU usage

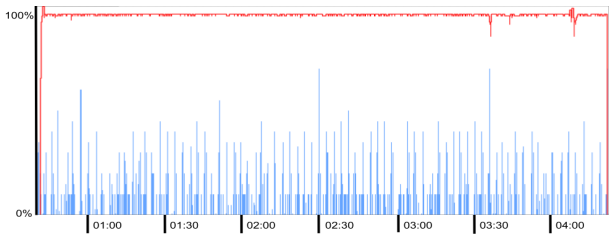

Figure 7: An Illustration of 'Normal' CPU Usage (Blue) and 'High' CPU Usage (Red)

The MetalKit graphic updates significantly contributes to the overall *Implementation Latency*, with 25.70% of this latency caused by the *CPU to Display Latency*.

The MetalKit *Input Latency* does not impact the *PVT Reaction Time* as significantly as the *CPU to display latency*, with only 0.50% of the latency attributed to this source.

3.2 UIKit Software Latency

It is expected that the UIKit average *Implementation Latency* will be greater than the MetalKit *Implementation Latency*, with a 33.04% increase over the ground truth RT. This is likely as it does not have the advantage of hardware acceleration that MetalKit provides [14]. The UIKit implementation displays the stimulus by setting the background colour of the UIView that the controller manages. Similarly to the MetalKit implementation, touches are handled in the view controller by overriding the *touchesBegan* method, which is reflected in the *Input Latency* in Table 1.

3.3 WebKit Software Latency

Finally, the WebKit implementation has the greatest average *Implementation Latency* with an increase of 48.58% over the ground truth RT. This may be due to using a *WKWebView* to render the HTML and execute the Javascript code, which adds an extra layer to the software stack. Latency increases are commonly caused by issues with the software stack [15].

4 EFFECTS OF DEVICE CONFIGURATION ON METALKIT IMPLEMENTATION

Mobile PVT suffers from a number of potential sources of variability. These include device CPU usage, Wi-Fi configuration, the Bluetooth configuration, mobile data status, battery percentage, power saving configuration, the OS version, background tasks or running applications. The effect of these sources may vary across different devices, resulting in the need to repeat these experiments on the participant's devices. It is also necessary to either standardize the device configuration during the PVT or to note the device configuration during the PVT. Using the ATI, it is possible to measure the effects of these sources caused by the device configuration.

In this paper, the effect of the device's CPU usage on the touch response latency was evaluated. The device was connected to the MacBook described in Section 2.3 for profiling. The CPU usage was analyzed using the Instruments application. The application was extended upon to include functionality to raise the CPU usage. This was achieved by executing an inefficient prime generating function in a thread with a high priority. The CPU usage for this experiment is outlined in Figure 7.

Using the ATI, 100 PVT trials ($n = 100$) with the CPU operating at a maximum of 70% usage were executed. The duration of this experiment was 5min 6s. The experiment was repeated with the CPU usage raised using the functionality described in Section 4. The CPU usage for this experiment is outlined in Figure 7. The duration of this experiment was 5min 16s. The results for both of these experiments are shown in Table 2. Using a MetalKit implementation of the PVT, there is a 3.01% increase in the average *PVT Reaction Time* with CPU usage at 99%.

5 DISCUSSION

We can conclude from our findings that the absolute RTs measured by mobile PVT are significantly overestimated. In this paper we demonstrated that the software implementation and CPU usage affect the absolute RT measurements. As discussed in Section 1, PVT metrics are based on absolute RTs. Although mobile PVT may be valid in assessing participant fatigue, it is concerning that potentially invalid RTs are used to analyse participant results.

For researchers currently using mobile PVT, we have provided an accessible method of measuring the latency and variability. Using the ATI, researchers can provide latency guarantees for any device(s) used to run their PVT implementation.

ACKNOWLEDGMENTS

This work is supported in part by Science Foundation Ireland (Grant No. SFI/12/RC/2289_P2) and AIB.

REFERENCES

[1] Saeed Abdullah, Elizabeth L. Murnane, Mark Matthews, Matthew Kay, Julie A. Kientz, Geri Gay, and Tanzeem Choudhury. 2016. Cognitive Rhythms: Unobtrusive and Continuous Sensing of Alertness Using a Mobile Phone. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany) *(UbiComp '16)*. Association for Computing Machinery, New York, NY, USA, 178–189. https://doi.org/10.1145/2971648.2971712

[2] Lucia Arsintescu, Jeffrey B. Mulligan, and Erin E. Flynn-Evans. 2017. Evaluation of a Psychomotor Vigilance Task for Touch Screen Devices. *Human Factors* 59, 4 (2017), 661–670. https://doi.org/10.1177/0018720816688394 arXiv:https://doi.org/10.1177/0018720816688394 PMID: 28095256.

[3] Mathias Basner and David F. Dinges. 2011. Maximizing Sensitivity of the Psychomotor Vigilance Test (PVT) to Sleep Loss. *Sleep* 34, 5 (05 2011), 581–591. https://doi.org/10.1093/sleep/34.5.581 arXiv:https://academic.oup.com/sleep/article-pdf/34/5/581/13665943/aasm.34.5.581.pdf

[4] Jean-François Brunet, Dominique Dagenais, Marc Therrien, Daniel Gartenberg, and Geneviève Forest. 2017. Validation of sleep-2-Peak: A smartphone application that can detect fatigue-related changes in reaction times during sleep deprivation. *Behavior Research Methods* 49, 4 (01 Aug 2017), 1460–1469. https://doi.org/10.3758/s13428-016-0802-5

[5] David F. Dinges, Frances Pack, Katherine Williams, Kelly A. Gillen, John W. Powell, Geoffrey E. Ott, Caitlin Aptowicz, and Allan I. Pack. 1997. Cumulative Sleepiness, Mood Disturbance, and Psychomotor Vigilance Performance Decrements During a Week of Sleep Restricted to 4–5 Hours per Night. *Sleep* 20, 4 (04 1997), 267–277. https://doi.org/10.1093/sleep/20.4.267 arXiv:https://academic.oup.com/sleep/article-pdf/20/4/267/8596499/sleep-20-4-267.pdf

[6] Sean P. A. Drummond, Amanda Bischoff-Grethe, David F. Dinges, Liat Ayalon, Sara C. Mednick, and M. J. Meloy. 2005. The Neural Basis of the Psychomotor Vigilance Task. *Sleep* 28, 9 (09 2005), 1059–1068. https://doi.org/10.1093/sleep/28.9.1059 arXiv:https://academic.oup.com/sleep/article-pdf/28/9/1059/8502897/sleep-28-9-1059.pdf

[7] Apple Inc. 2020. *Metal*. Apple Inc. https://web.archive.org/web/20200422010052/https://developer.apple.com/documentation/metal/

[8] Apple Inc. 2021. *Instruments Help*. Apple Inc. https://web.archive.org/web/20210610195056/https://help.apple.com/instruments/mac/current/

[9] Aditya Jain, Ramta Bansal, Avnish Kumar, and K. D. Singh. 2015. A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students. *International journal of applied & basic medical research* 5, 2 (2015), 124–127. https://doi.org/10.4103/2229-516X.157168

[10] Matthew Kay, Kyle Rector, Sunny Consolvo, Ben Greenstein, Jacob O. Wobbrock, Nathaniel F. Watson, and Julie A. Kientz. 2013. PVT-touch: Adapting a reaction time test for touchscreen devices. In *2013 7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*. 248–251. https://doi.org/10.4108/icst.pervasivehealth.2013.252078

[11] Maxim Y. Khitrov, Srinivas Laxminarayan, David Thorsley, Sridhar Ramakrishnan, Srinivasan Rajaraman, Nancy J. Wesensten, and Jaques Reifman. 2014. PC-PVT: A platform for psychomotor vigilance task testing, analysis, and prediction. *Behavior Research Methods* 46, 1 (01 Mar 2014), 140–147. https://doi.org/10.3758/s13428-013-0339-9

[12] Julian Lim and David F. Dinges. 2008. Sleep deprivation and vigilant attention. *Annals of the New York Academy of Sciences* 1129 (2008), 305–322. https://doi.org/10.1196/annals.1417.002

[13] Sylvia Loh, Nicole Lamond, Jill Dorrian, Gregory Roach, and Drew Dawson. 2004. The validity of psychomotor vigilance tasks of less than 10-minute duration. *Behavior Research Methods, Instruments, & Computers* 36, 2 (01 May 2004), 339–346. https://doi.org/10.3758/BF03195580

[14] Albert Ng, Julian Lepinski, Daniel Wigdor, Steven Sanders, and Paul Dietz. 2012. Designing for Low-Latency Direct-Touch Input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 453–464. https://doi.org/10.1145/2380116.2380174

[15] Tom Pantels, Sheng Guo, and Rajshree Chabukswar. 2014. Touch Response Measurement, Analysis, and Optimization for Windows* Applications.

[16] Edward Price, George Moore, Leo Galway, and Mark Linden. 2017. Validation of a Smartphone-Based Approach to In Situ Cognitive Fatigue Assessment. *JMIR Mhealth Uhealth* 5, 8 (17 Aug 2017), e125. https://doi.org/10.2196/mhealth.6333

[17] Thomas Pronk, Reinout W. Wiers, Bert Molenkamp, and Jaap Murre. 2020. Mental chronometry in the pocket? Timing accuracy of web applications on touchscreen and keyboard devices. *Behavior Research Methods* 52, 3 (01 Jun 2020), 1371–1382. https://doi.org/10.3758/s13428-019-01321-2

[18] Benjamin Tag, Andrew W. Vargo, Aman Gupta, George Chernyshov, Kai Kunze, and Tilman Dingler. 2019. *Continuous Alertness Assessments: Using EOG Glasses to Unobtrusively Monitor Fatigue Levels In-The-Wild*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300694