Draining our Glass: An Energy and Heat Characterization of Google Glass

Technical Report 2014-3-23, Rice University

Robert LiKamWa[†], Zhen Wang[†], Aaron Carroll[†‡], Felix Xiaozhu Lin[†], and Lin Zhong[†]

[†]Rice University, Houston, TX [‡]UNSW, Australia

ABSTRACT

The Google Glass is a mobile device designed to be worn as eyeglasses. This form factor enables new usage possibilities, such as hands-free video chats and instant web search. However, its shape also hampers its potential: (1) battery size, and therefore lifetime, is limited by a need for the device to be lightweight, and (2) high-power processing leads to significant heat, which should be limited, due to the Glass' compact form factor and close proximity to the user's skin. We use the Glass in a case study of the power and thermal characteristics of optical head-mounted display devices. We share insights and implications to limit power consumption to increase the safety and utility of head-mounted devices.

1. INTRODUCTION

Optical Head Mounted Display (OHMD) devices, including [4, 24], provide users with a hands-free display, with rich user-centric experiences and immediate access to computing resources. Recently, interest has drawn to Google's spectacle-shaped device, called *Glass*, marking a commercial advancement in wearables, shown in Fig. 1.

In this article, we share results from characterizing Glass' power draw. Others have documented other aspects, e.g., technical specifications [20] and privacy, security, and social concerns [9, 21]. In contrast, we study Glass as an OHMD system, especially the power draw of its components and the form factor's implication on app usage and system design.

While it is tempting to treat the Glass architecture as a smartphone or tablet in a different form factor with new use cases, OHMD physical limitations magnify the value of efficiency, as compactness limits battery capacity. Moreover, contact with a user's skin will make heat generation from power draw uncomfortable and potentially dangerous.

Thus, as low-power constraints pose the greatest technical challenge to OHMDs, we perform component-driven and usage-driven power analysis. We find that many scenarios, including mobile vision and long-term video chats, are not possible under the system's power draw. Following our analysis, we discuss hardware and software insights regarding heat constraints, display efficiency, and processor power that motivate study into efficient OHMD system design.

2. GLASS SYSTEM OVERVIEW

Google's Glass system resembles smartphone architectures with a few notable differences: there is no cellular modem; the display is much smaller; the touchpad is distinct from the display; and a bone-conduction speaker provides audio output. Many other components can be found on smartphones.

The design centers around an OMAP4430 system-on-chip (SoC), which includes a dual-core ARM Cortex-A9 as the main CPU, a dual-core ARM Cortex-M3, an SGX540 GPU, a Display Subsystem (DSS), an Image and Video Accelerator (IVA) and a DSP [22]. The OMAP4430 is also used in the Motorola Droid RAZR, the LG Optimus 3D Max, the Samsung Galaxy Tab 2, and many other mobile devices.

Because of the OHMD form factor, the OLED or LCD smartphone display is absent, replaced with a Liquid-Crystal-on-Silicon (LCOS) projection display, shown in Fig. 2. As an LED is filtered to shine red, green, or blue light onto the LCOS, the display shows the respective color component contribution. The image is projected on a semi-reflective mirror directly in front of the user's right eye. This allows content to be seen not only by the user, but also from the front of Glass. As red, green, and blue images are flashed rapidly, the user perceives a full-color image. According to [1], component images must cycle at 540 Hz for color video.

Glass employs a unique bone-conduction speaker to allow the user to hear clearly while minimizing the sound to people not using the device. A Synaptics touchpad component provides touch functionality on the side of the Glass.

A 3.7 volt LiPo battery with a capacity of 2.1 Wh provides power to the Glass. The battery sits behind the ear, counterbalancing the rest of the device over the user's ear.

Glass runs Android OS, v. 4.04 "Ice Cream Sandwich".

3. USE CASES

As a hands-free device with display, camera, and radios, Glass creates new usage potential. While not an exhaustive list, we discuss potentially useful OHMD apps. Most are advertised by Google and are marked with * [6].

Real-time Hands-free Notifications*: Users can field phone calls, text messages, e-mails, etc., without reaching into a pocket or bag. This mitigates user disruptions, e.g., a user can ride a bike while addressing a text message.

Hands-free visual and audio instructions*: OHMDs can access instructions and materials related to a user's activity. This is useful for a cook following a recipe, an artist consulting reference material, or a tourist asking for language translation. OHMDs could even assist surgeons using CT or X-ray images during surgery [12].

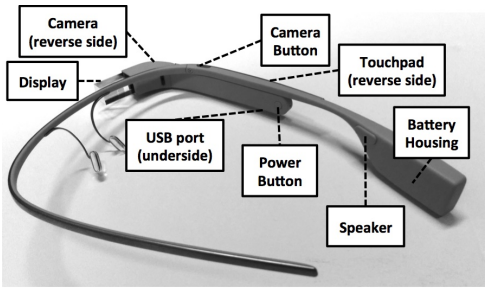

Figure 1: Google Glass user interface hardware

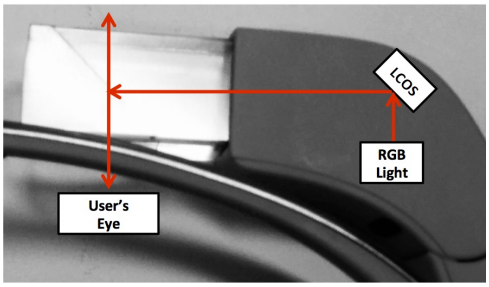

Figure 2: Google Glass display projection path

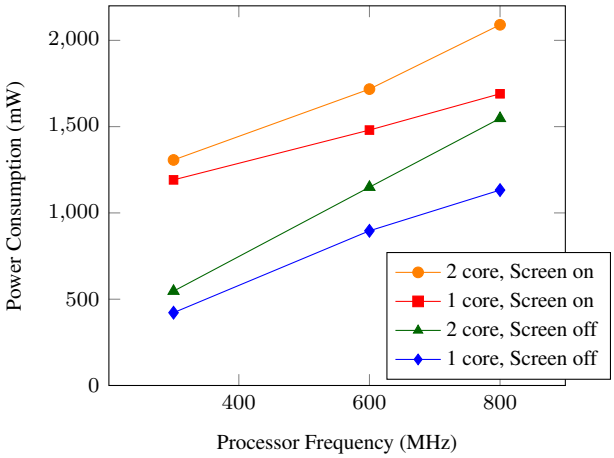

Figure 3: Power consumption vs. core use, with one core and two cores at 100% CPU Utilization, with screen on and screen off

Instant Connectivity Access*: Users can instantly access network-based information, including email inboxes, web searches, local news, stock market tickers, home security camera feeds, or social networks.

Instant Photography/Videography*: OHMD cameras allow users to take pictures and video clips without pointing a smartphone. This is useful for journaling momentary experiences or recording a user's perspective of a scene. Video can also be streamed in real-time for engaging video chats.

Augmented Reality*: In augmented reality (AR), virtual objects are displayed as part of the physical environment. AR can overlay path directions, assist in interior design, or enable other immersive visualization apps. The Glass Developer documentation lists ideas for simple AR games which use voice and inertial motion sensors for interaction [5].

Continuous Mobile Vision: A technology that many expect out of OHMDs is the ability to observe and understand a scene through computer vision. Face detection and recognition are feasible, while discouraged by Google. Vision can be used for many other tasks, including text recognition, geometric scene understanding, and contextual life logging.

These scenarios would provide a rich experience to the OHMD user. We benchmark representative workloads in Section 4. These potential use cases dictate device requirements and thus influence the system architectural design.

4. POWER MEASUREMENT

We benchmark the power draw and CPU usage. To measure power consumption, we use a Monsoon Power Monitor [16] in place of the Glass battery. The monitor provides 4 V and records power draw. For CPU utilization, we use the command `top`, which has a 5–7% CPU utilization overhead.

4.1 Power by component

We first explore how a component's use contributes to overall system power. We configure and utilize each component while keeping the rest of the system constant. Our measurements thus reveal the *rise in system power consumption* while a component is used.

OMAP4430 SoC: The OMAP is a major contributor to Glass' power consumption. We check the status of its modules by using the `omapconf` [23] diagnostic tool. By default, many modules are disabled even when the screen is on, including the Cortex-M3, GPU, IVA and DSP. This leaves the Cortex-A9 as the major active computational component.

Unlike smartphones, the Glass uses the DSS instead of the GPU to perform surface composition and to stream frames to the display. Thus, the DSS is on while the screen is on, and the GPU is disabled by default. The GPU may be invoked by software, e.g., for OpenGL applications. The IVA is activated for encoding/decoding video recording and playback.

The main CPU (the Cortex-A9) can be set to four frequencies: 300 MHz, 600 MHz, 800 MHz, and 1 GHz. Raised frequencies increase performance, but draw more power. At high temperatures, Glass firmware limits the frequency to 600 MHz or 300 MHz to cool down by reducing power. We run shell scripts on one Cortex-A9 core and both cores at 100% CPU utilization with the screen on and off. We then measure the power draw of the Glass system, shown in Fig. 3. While we can briefly set the Glass to 1 GHz, the system rigorously decreases the frequency to reduce heat, prohibiting us from taking robust 1 GHz measurements.

Screen: Glass sets the screen brightness on a 25–255 scale depending on the sensed ambient brightness. We set the brightness by writing to a device file while using a static app with static screen content. As shown in Fig. 4, the brightness affects the Glass' power consumption. The screen content, including its colors, does not affect the power draw. This is similar to LCDs but in contrast to OLED displays.

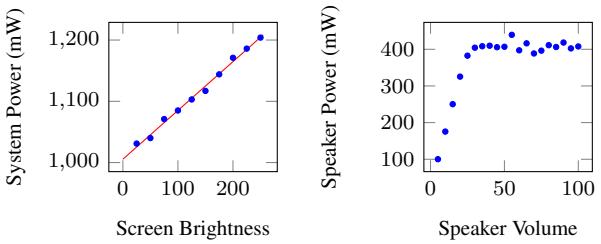

Figure 4: (Left) System power draw vs. screen brightness. (Right) Speaker power draw vs. speaker volume.

We measure that Glass draws 1028 mW when the screen is at a brightness of 25. Glass draws 1204 mW at a brightness of 255. By contrast, with the screen off, but the system active, the Glass draws 334 mW. Thus, when using the screen, the system draws a static power of 674 mW and dynamically draws another 196 mW depending on the brightness level.

$$P_{\text{screen}} = 674\,\text{mW} + 196\,\text{mW} \times (\text{brightness}/255)$$

The high static power draw of 674 mW is likely due to the activation of the DSS display subsystem, its rendering of the screen content, and the transmission to the LCOS.

As the display is close to the eye, we expected the dynamic power to be orders of magnitude lower than that of a smartphone. Display power is typically proportional to D^2 where D is the distance from the screen to the eyes, if all other factors remain the same [25]. Since D for the Glass is over 10× smaller than that for a phone, we expected a display draw of < 5 mW, as the iPhone 4 LCD consumes ∼420 mW [10] at full brightness. The actual projection consumes up to 196 mW, much higher than expected. This is likely due to luminance drops in the display path. First, the color filter reduces the luminance of the LED projection. Luminance is further reduced by ∼40% in reflection off of typical LCOS devices [8]. Finally, the semi-reflective mirror in front of the user's eye incurs a >50% drop in the reflective optics. These drops necessitate high-power projection at the source.

We perform the rest of our measurements with the screen brightness fixed to 25, suitable for our office environment.

Bone-Conduction Speaker: Using the bone-conduction speaker consumes ∼410 mW when the volume is at or above 35%. The sound production is not louder above 35%. Below 35%, power consumption decreases, down to 100 mW at 5%. When playing audio through a USB Earpiece, Glass draws 18–30 mW. For comparison, we measure that a Galaxy Nexus draws ∼200 mW when using its speakers.

Inertial Motion Unit: The Android API can sample the accelerometer and gyroscope at either 100 Hz or 200 Hz. Sampling consumes ∼29 mW, regardless of sampling speed.

Audio Recording: Using the microphone on the Glass to record audio consumes an additional 96 mW.

WiFi/Bluetooth: We measure the Glass before activating WiFi/Bluetooth and during a file download. While downloading at 538 kbps over Bluetooth, Glass draws an additional 743 mW. On WiFi, at 734 kbps, Glass draws 653 mW.

Table 1: Glass power draw in different usage scenarios

Usage Case	Power Consumption	Battery Life
Idle	22 mW	95 hours
System active, screen off	334 mW	377 min.
Static timeline card	1030 mW	122 min.
Timeline swiping	1315 mW	96 min.
"Ok Glass" card	1204 mW	105 min.
Main menu card	2361 mW	53 min.
Internet page load	2009 mW (3 sec.)	1260 page loads
Web page viewing	1171 mW	107 min.
Web page scrolling	1505 mW	84 min.
Phone calls	1257 mW	100 min.
Text message	1387 mW (1.3 sec.)	4200 messages
Image capture	2927 mW (3.3 sec.)	782 images
Video capture	2963 mW	43 min.
Video chat	2960 mW	43 min.
Static application	1023 mW	123 min.
Camera preview callback	2366 mW	53 min.
OpenCV face detection	3318 mW	38 min.

4.2 Power by Usage Scenario

We next collect average power measurements as we place the Glass under various app workloads, as shown in Table 1.

Idle Power: Glass exhibits an efficient idle mode, using background processing to sense wake-up events, such as notifications or accelerometer gestures. Processors, radios, and the display are held in an inactive low-power mode. The idle Glass consumes 22.3 mW for a lifetime of over 90 hours.

When the system wakes up, the CPU briefly rises to 1 GHz to quickly wake the system up, and subsequently returns to 300 MHz, waiting for user interaction. This wake-up process takes 1.9 seconds and consumes 1398 mW on average.

Menu Navigation: After waking up, Glass opens a heads-up menu. Primary interaction is through the touchpad: the user swipes to highlight items, and taps to activate one. This lets the user access instant connectivity, address notifications, or initiate hands-free visual and audio interactions.

When the user is observing a static timeline card, the CPU utilization is near 0% and the Glass draws ∼1030 mW, giving the Glass a battery lifetime of 2 hours in this scenario. When the user is swiping through the cards, a "mediaserver" process is invoked requiring approximately 8–10% CPU utilization at 600 or 800 MHz, to load the new processes. The system draws additional power to a total of 1315 mW while being swiped, but this only lasts as long as the swipe persists.

Voice command is available on the main menu card, where the Glass displays the time. The system waits for a user voice input and consumes 1204 mW, which is about 200 mW more than static timeline cards due to the background voice recognition process with microphone sampling. When the keyphrase "OK Glass" is spoken, a submenu of speakable items appears. A user selects an item for launch by using the touchpad or reading its title. Glass draws 2361 mW while the user is navigates the menu by voice. Actively recognizing voice uses 30% CPU utilization at 300 MHz for the voice recognition process, and samples the microphone sensor.

Internet Browsing: The built-in internet browser application does not have an address bar to enter a URL, but a user can perform a Google voice search and access the web page result. Once in a webpage, the user places two fingers on the touchpad and moves their head to pan a website.

Loading a Wikipedia page takes ∼3 seconds, incurs CPU utilization of 50%, and draws 2009 mW. Viewing the loaded page draws 1171 mW, while scrolling the page draws 1505 mW. This is comparable to browsing on a Galaxy S III [3]. As with the smartphone, the high power draw is mainly due to the high static power of using the display.

Telephony: Glass can operate as a headset, routing phone call audio between the phone and the Glass for hands-free communication. This uses the microphone, speaker, and the Bluetooth card. Running the telephony incurs a 35% CPU utilization and draws 1257 mW, for a battery life of 102 minutes. Receiving a text message draws 1387 mW for 1.3 seconds. 4200 messages can be received on a single charge.

This is disproportionately high; a Bluetooth headset consumes an order of magnitude less power. This is likely because the entire Glass system is awake and using the high-power Cortex-A9 to perform simple telephony tasks.

Image/Video Capture and Streaming: Glass can instantly capture photos and videos on demand. Unfortunately, the image capture and processing draw significant power. Using the built-in camera app to take a picture consumes a peak of 4629 mW and an average of 2927 mW for 3.3 seconds. Glass takes fewer than 800 pictures on a single charge.

For video recording, our measurements show that the Glass draws 2963 mW while using the camera app. At that power draw, the Glass can take a video for 45 minutes on a single battery charge. Streaming a video call over WiFi draws roughly the same amount of power, consuming an average of 2960 mW, allowing a 45 minute video cat on a single charge.

Vision Application Usage: Developers may write applications using the standard Android SDK and deploy them on the Google Glass. Running a static application draws similar power to that of a static timeline card: 1023 mW. In this scenario, the system uses around 18% CPU utilization.

Glass vision apps can be developed by using the preview frame callback of the Android camera service. This is a significant workload for the Glass; the act of running the camera service and previewing a frame uses 85% CPU utilization at 600 MHz. Furthermore, the IVA and GPU are activated. As a result, the system draws a total power of 2366 mW.

To operate on the frame, we use the OpenCV library with Android bindings. Running a Face Detection app brings the device to a full 100% CPU Utilization at 600 MHz. This draws a large 3318 mW, for only 38 minutes of battery life.

5. TEMPERATURE CHARACTERIZATION

As with all electronics, power is linked to temperature, as nearly all power is dissipated as heat. Because Glass makes contact with its user's face, heat is a critical safety issue.

Measurement

To characterize thermal behavior, we use an ST-380 surface thermometer pointed where Glass makes contact with the user's temple, the face region behind the eyes. This tends to be the warmest part of Glass, as it is where the OMAP is located. The OMAP is the major element that contributes to the temperature in the measured area; changing screen brightness or using the speaker did not change the reported temperature.

Unlike our power measurements, we used a Glass that we did not tamper with. The room was held at 23°C with adequate ventilation, and the Glass is 31.1°C while charging. After disconnecting the charger, we started a video chat. The thermometer reported a rapid rise, shown in Fig. 5. After 120 seconds, the Glass rose to 39°C. The rise eventually slowed, reaching a stable 51.9°C (124°F). We repeated the process with a static idle app, reaching a stable 35.2°C.

Modeling

We next use Newton's Cooling Law [2] to relate SoC power draw to the steady-state temperature difference (ΔT) of the Glass with its environment. We model ΔT as proportional to the power dissipation (P) and inversely proportional to the surface area (A) and the convection coefficient (h). The initial device temperature does not affect ΔT.

$$P = dQ/dt = hA\Delta T$$

Newton's Cooling Law also models how temperature changes over time to reach the steady-state temperature. For a device of heat capacity C, the temperature at a given time is:

$$T(t) = T(0) + (1 - e^{-(C/(hA))t})\Delta T$$

Thus, power consumption of the device dictates the steady-state temperature difference (ΔT); how fast the temperature rises is determined by the mechanical and material properties of the device and its environment, i.e., C, h, and A.

We fit the model to our data in Fig. 5. While the OMAP consumes 3 watts during video chat, we model that $\Delta T = 28.9°C$, and $C/hA = 0.0040\,s^{-1}$. For 1 watt draw in static app usage, we model that $\Delta T = 11.2°C$, and $C/hA = 0.0040\,s^{-1}$. Thus, we have the following observations for Glass when used in a similar office environment to ours:

- For every watt drawn by the OMAP, the steady-state surface temperature will rise by $\Delta T \approx 10°\text{C}$ higher than its environment.
- The Glass surface temperature will rise by 90% of ΔT in 10 minutes.

That is, the temperature difference from the environment is roughly proportional to the SoC's power draw, while the rate of heating is dependent only on its environment.

Thus, *long-term average power consumption of a process dictates stable temperature and should be constrained for heat safety*. This may incur energy inefficiency, as the time

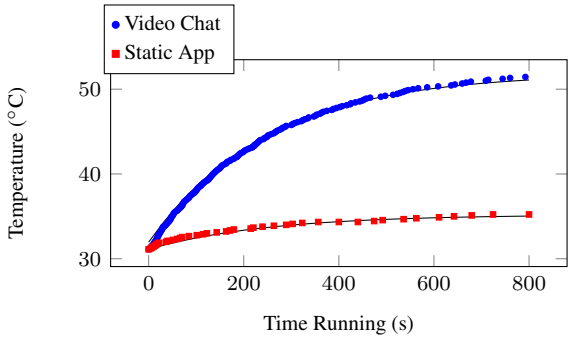

Figure 5: Temperature samples vs. time running an application

it takes to run a low-power process may exceed the optimal point to minimize energy use. However, constraining average power will keep the device temperature in safe limits.

We collect measurements while the Glass is not worn by a user. A user may raise the Glass' temperature due to the user's body being warmer than that of the environment. Indeed, the authors experience that the device feels hot while performing intensive tasks. The Glass could also experience an even higher temperature with a harsh environment, e.g., in direct sunlight in 38°C weather.

Health Implications

High temperatures put pressure on the human body to regulate its temperature to below 37°C, which it does by dilating blood vessels, increasing heart rate, raising skin temperature, and activating sweat glands, as explained in [11]. These reactions lead to reduced comfort and potential cardiovascular problems. Indeed, blood vessel damage can occur with continuous contact to surfaces at temperatures as low as 38–48°C , leading to permanent skin damage, such as *erythema ab igne* [19]. Thus, while the measured surface temperatures are common for smartphones and tablets, the device surface temperatures of 50+°C are not well-suited for a head-mounted device with large durations of skin contact.

6. INSIGHTS AND RECOMMENDATIONS

Glass consumes an amount of energy close to that of a smartphone. This, paired with skin contact with the user, places thermal issues as a first-class design constraint. The high power draw also limits usage time due to the small battery size. Among the use cases in Section 3, *only hands-free notifications and instant connectivity are safe and feasible*.

Long-term visuals and audio are constrained by the power draw of using the display and speaker. Photography and video chats are further limited; camera usage draws ∼3 W, heating Glass 28°C above its environment. While the form factor suggests easy image capture and vision opportunities, power draw limits the longevity and safety of such apps.

Thus, the Glass design is not suitable for always-on use, including long-term AR and mobile vision. While off-loading perception tasks to the cloud is difficult, due to transmission cost and need for connectivity [7], onloading computations is impractical due to high power draw. For these tasks, a smartphone-like system cannot achieve the required efficiency. Instead, we propose the following considerations.

Display Efficiency

In addition to computational power expense, the system consumes an additional 675 mW to show screen content, and up to another 200 mW depending on the brightness. This prohibits long-term use of the screen. We recommend an adoption of efficient content generation and projection.

Static Display Subsystem: While Glass should be able to display active scenes, the DSS consumes too much for slow-changing content, e.g., displaying a cookbook. One potential solution could be to use a hybrid DSS with a bistable mode to provide static content for low-power viewing.

A hybrid DSS would introduce system research challenges of optimally deciding which mode to use, as well as supporting migration between active and static DSS modes.

Efficient Projection: The LCOS projection consumes two orders of magnitude more power than its proximity requires, due to luminance drops in the display path. Replacing this projection with a transparent OLED screen in front of the eye would reduce power draw by avoiding luminance drops. Moreover, the see-through nature of OHMDs means that for many apps, few pixels are active at any time. This is optimal for OLED, as inactive pixels do not draw power.

Computational Efficiency

Computational power draw is the major contributor to Glass' high power consumption. This strongly motivates principles of energy-proportionality and optimizing the common case.

Heterogeneous Computing: OHMDs present conflicting processor requirements: always-on cases demand high efficiency at low throughput, medium-term power is limited by battery life and device heat, and user I/O demands intermittent high throughput to limit visible latency. While the Cortex-A9 provides high throughput, merely onlining the CPU consumes ∼330 mW, making it unsuitable for always-on cases. Core heterogeneity is a known solution to this problem. We believe this is not merely convenient for longer battery life, but in fact required for the OHMD form factor to deliver on its promised functionality. System support for core heterogeneity is an active area of research [14, 15], and our results provide a further data point in motivating this work.

Vision Acceleration: Vision processing on the CPU draws a high 3.3 watts on the Glass, limiting the use cases for readily-available camera capture. For these apps to be feasible, an efficient vision processing unit, such as the Movidius Myriad [17], will be needed. Existing application-specific circuity, such as GPUs and video codecs, have well-defined interfaces (OpenGL, MPEG, etc.). We see a research opportunity in exposing vision acceleration to user apps, including issues of resource management, multiplexing, isolation, etc.

Responsible Thermal Control

Because of user contact, heat is a paramount concern, as every watt generates a 10°C rise. While lowering power draw is necessary to reduce heat, optimizing heat dissipation and introducing system policies can alleviate thermal issues.

Improved Heat Dissipation: Heat dissipation can be partially addressed with physical redesign. To increase user comfort and safety, the Glass form factor should distance processing elements from skin contact and use heat sinks to dissipate OMAP heat over a wider surface area.

Thermal Regulation: Thermal regulation is well-studied, usually through limiting power draw to the thermal design point (TDP). We have shown that power should be constrained to a safe user limit much lower than typical TDP. Moreover, Glass exceeds this limit under typical operation. Thus, the system should be designed for graceful degradation. For instance, when the thermal limit is reached during video chat, the resolution, frame rate, encoding quality, bitrate, etc. *must* be reduced to avoid physical harm. Typical approaches of reducing frequency, disabling cores, or killing processes are not satisfactory, as this situation occurs in everyday usage.

Additionally, thermal limits are not hard, i.e., peak power need not be strictly limited. As Fig. 5 shows, the heating time constant is relatively long. Efficient thermal regulation should control average and not peak power, but it must also be aware of the time period over which this average must be maintained. Strategies such as [18] can computationally sprint to briefly raise peak power while managing heat.

Low Power I/O

The camera, screen, and speaker are designed for high power, high quality use. Glass should have I/O quality mode options for power reduction strategies.

Scalable Imaging: Using the camera reduces battery life to one hour due to the high-power IVA and image sensor, regardless of capture quality. Image sensor energy cost should be proportional to the frame rate and resolution, using techniques such as [13]. Also, a multi-rate IVA could provide low-power processing when high resolution is not required.

Notification LED: When a text message arrives, the DSS is activated, consuming precious energy. Instead, an LED on the display could notify the user, to avoid display activation. This is an effective strategy for smartphone notifications, and should be adopted for OHMD energy efficiency.

Moving Coil Speaker: The bone-conduction speaker is energy expensive for continuous use, such as for music or video playback or for game audio. A supplementary moving coil speaker would allow a choice for lower-power audio.

7. CONCLUSION

We use Google Glass as a platform for studying system aspects of wearable devices that are constrained by form factor and battery life. While it has suboptimal power draw and heat dissipation, it provides an exciting public introduction to wearable devices and a base for future OHMD design.

The Glass device motivates deep investigation into wearable systems. We find that the high performance, significant power draw, and thermal concerns present several OHMD research opportunities towards improved efficiency.

8. REFERENCES

[1] M. S. Brennesholtz and E. H. Stupp. *Projection Displays*. Wiley Series in Display Technology. Wiley, 2008.

[2] Louis C. Burmeister. *Convective Heat Transfer*. A Wiley-Interscience publication. Wiley, 1993.

[3] A. Carroll and G. Heiser. The systems hacker's guide to the galaxy energy usage in a modern smartphone. In *Proc. Asia-Pacific Workshop on Systems (APSys)*, 2013.

[4] Epson. Moverio Smart Glasses. `http://epson.com/cgi-bin/Store/jsp/Moverio/Home.do`.

[5] Google. Glass Developers - Mini Games. `https://developers.google.com/glass/samples/mini-games`.

[6] Google. How it Feels. `http://google.com/glass/start/how-it-feels/`.

[7] S. Han and M. Philipose. The case for onloading continuous high-datarate perception to the phone. In *HotOS*, 2013.

[8] HoloEye. HoloEye Microdisplay Technology Brochure. `http://holoeye.com/wp-content/uploads/2011/07/LCOS_Microdisplays.pdf`.

[9] J. Hong. Considering privacy issues in the context of google glass. *Commun. ACM*, 56(11):10–11, November 2013.

[10] Khronos Group. DisplayMate iPhone 4 ShootOut. `http://displaymate.com/iPhone_4_ShootOut.htm`.

[11] K. H. E. Kroemer, H.B. Kroemer, and K.E. Kroemer-Elbert. *Ergonomics: how to design for ease and efficiency*. Prentice-Hall int'l series in industrial and systems engineering. Prentice Hall, 1994.

[12] L. Kim. Google Glass Delivers New Insight During Surgery. `http://ucsf.edu/news/2013/10/109526/surgeon-improves-safety-efficiency-operating-room-google-glass`.

[13] R. LiKamWa, B. Priyantha, M. Philipose, L. Zhong, and P. Bahl. Energy characterization and optimization of image sensing toward continuous mobile vision. In *Proc. ACM Int. Conf. Mobile Systems, Applications, & Services (MobiSys)*, 2013.

[14] F. X. Lin, Z. Wang, R. LiKamWa, and L. Zhong. Reflex: using low-power processors in smartphones without knowing them. In *Proc. ACM Int. Conf. Architectural Support for Programming Languages & Operating Systems (ASPLOS)*, pages 13–24, 2012.

[15] F. X. Lin, Z. Wang, and L. Zhong. K2: A mobile operating system for heterogeneous coherence domains. In *Proc. ACM Int. Conf. Architectural Support for Programming Languages & Operating Systems (ASPLOS)*, 2014.

[16] Monsoon. Monsoon power monitor. `http://msoon.com/LabEquipment/PowerMonitor`.

[17] Movidius. Movidius Myriad 1. `http://movidius.com/technology/myriad1/`.

[18] A. Raghavan, L. Emurian, L. Shao, M. Papaefthymiou, K. Pipe, T. Wenisch, and M. Martin. Computational sprinting on a hardware/software testbed. In *Proc. ACM Int. Conf. Architectural Support for Programming Languages & Operating Systems (ASPLOS)*, 2013.

[19] R. R. Riahi and P. R. Cohen. Laptop-induced erythema ab igne: report and review of literature. *Dermatology online journal*, 18(6), 2012.

[20] S. Torborg and S. Simpson. Google Glass Teardown. `http://catwig.com/google-glass-teardown/`.

[21] T. Stevens. Google Glass review (Explorer Edition). `http://engadget.com/2013/04/30/google-glass-review/`.

[22] Texas Instruments. OMAP4430. `http://ti.com/product/OMAP4430`.

[23] Texas Instruments. OMAPCONF: Texas Instruments OMAP Processors Diagnostic Tool. `https://github.com/omapconf/omapconf`.

[24] Vuzix. `http://vuzix.com/`.

[25] L. Zhong and N.K. Jha. Energy efficiency of handheld computer interfaces: limits, characterization and practice. In *Proc. ACM Int. Conf. on Mobile systems, applications, and services (MobiSys)*, 2005.