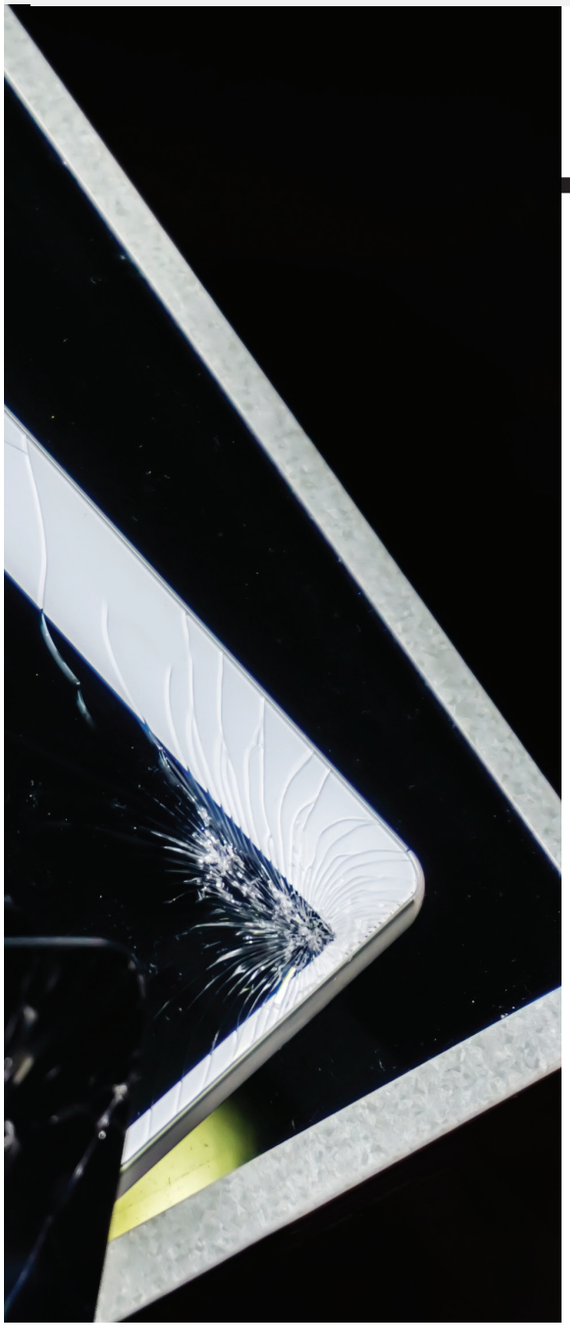
# YOU CAN'T TEACH A NEW PHONE OLD TRICKS: Smartphones Resist Traditional Compute Roles

Excerpted from "Experience: Android Resists Liberation from Its Primary Use Case" from MobiCom '18, *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* with permission. https://dl.acm.org/citation.cfm?id=3241583 © ACM 2018

The smartphone is an incredible computing platform. Loaded with powerful processing, vast data storage, near-global connectivity, built-in batteries, and a rich array of sensors, these devices reliably service the needs of billions of users every day. However, when tasked to run just a single application continuously without any human interaction, the smartphone platform becomes surprisingly unreliable. Over the course of a four-month deployment of Android-phone-based cellular gateways in Zanzibar, Tanzania, all 16 deployed phones failed despite significant engineering efforts, and six phones became physically damaged. This article examines what went wrong and how mobile computing platforms could adapt to support more traditional embedded computing roles and workloads.

Photo, istockphoto.com



**Noah Klugman, Meghan Clark, Matthew Podolsky and Pat Pannuto** *University of California, Berkeley, Berkeley, CA*
**Jay Taneja** *University of Massachusetts, Amherst*
**Prabal Dutta** *University of California, Berkeley, Berkeley, CA*

**Editors: Nic Lane and Xia Zhou**

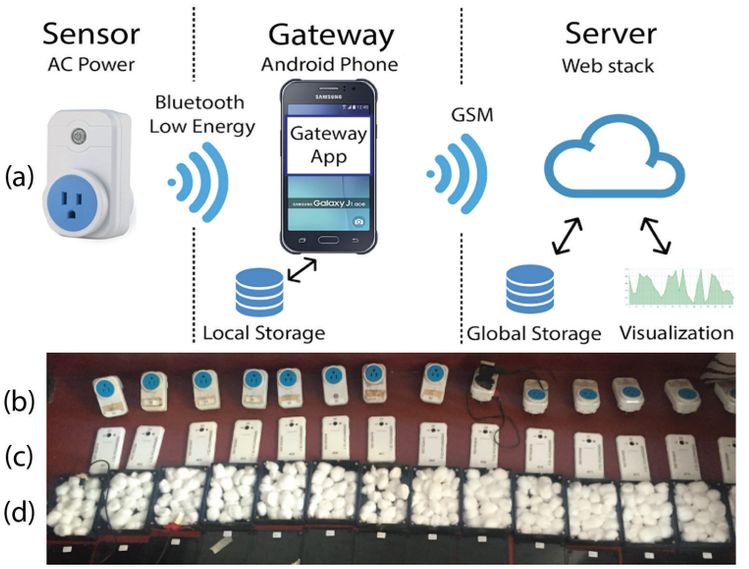



**FIGURE 1.** PlugWatch system architecture and implementation. A gateway app (a) on an unmodified Android phone (Samsung J1 Ace running Android 4.4.4) connects to a commercial plug-load power meter (WiTenergy E110) over Bluetooth Low Energy and forwards the power data from the sensor to a cloud server. In our implementation, the plug-load meter (b) plugs into the wall, and the phone (c) plugs into the meter. The phone is placed in a plastic box (d) which is padded with cotton balls and screwed shut.

This small deployment should have been straightforward. Collaborators exploring electricity grid reliability wanted to measure voltage quality for a small number of households in rural Tanzania. There are a wealth of existing commercial "smart plug" power meters marketed for smart home applications, but these sensors need to offload data via nearby Wi-Fi or Bluetooth, which was unavailable in participant homes. We turned to the smartphone, thinking it would be simple to write a gateway app to forward data from Bluetooth over a cellular connection back to our servers.

Unfortunately, our experience suggests that many of the expected benefits of building a user-free, phone-based gateway are either unattainable or difficult to achieve in practice. Our work inadvertently revealed the issues that arise from the fundamental tension of converting a general-purpose, user-facing platform into a single-purpose, remotely supervised device.

## DESIGN CONSIDERATIONS

Figure 1 presents the PlugWatch system architecture. PlugWatch uses a plug-load power meter as a sensor and forwards data via an Android smartphone. The decision to build around a smartphone was made with several expected benefits:

- **Reduced Development Time:** By using a smartphone as the primary processor, local database, and radio interface, we expected to reduce development time and costs compared to designing custom hardware.
- **Increased Reliability:** Android and Google services, such as the Play Store, offer over-the-air updates and remote monitoring out of the box.
- **Easier Debugging:** The mature development tools and built-in GUI on a smartphone help with debugging and verifying local network connectivity.

## REQUIREMENTS

The design of the PlugWatch system met a number of functional requirements:

- **Minimal Human Interaction:** We decided to keep humans out of the loop as much as possible to reduce the burden of participation and remove noise from incorrect human operator actions.

- **Reliably Long-Running:** Our experiment was intended to measure variance in voltage and power quality for many months, so we designed PlugWatch to run 24 hours a day.
- **Non-Obtrusive:** Since the deployment sites were households, we designed PlugWatch and configured Android to not make noise, display lights, or otherwise annoy participants.
- **Real-Time Data Stream:** Our utility partners were interested in receiving plug-load information in real time to experiment with how they might process smart meter datastreams.
- **Low Budget:** Our budget was modest and targeted a complete per-unit system cost of approximately $100 USD. This partially motivated our selection of an economical mid-range phone available in the local geography.

## IMPLEMENTATION CHALLENGES

PlugWatch was developed and tested in the lab, then assembled in the field, as shown in Figure 1. Actually repurposing smartphones to operate independently from human contact revealed a number of challenges.

### Staying On

Despite its large battery, a phone may still completely exhaust its energy reserves during long power outages and then shut down. Most phones are configured to charge – but not power on – after an outage. Indeed, only select phone models even have the hardware to support automatic power-on, and achieving this functionality required rooting the phone and exploiting a hack that replaces the charging image to boot the phone.

### Staying Connected

Staying connected to the cellular network requires manual in-country maintenance. The local carrier offered prepaid data plans in which airtime is purchased as a code behind a scratch-off ticket. This code must be entered by hand into the phone over an Unstructured Supplementary Service Data (USSD) interface, which is only available in-country. The purchased bandwidth expires regardless of usage rates on timelines ranging from a few hours to 60 days.

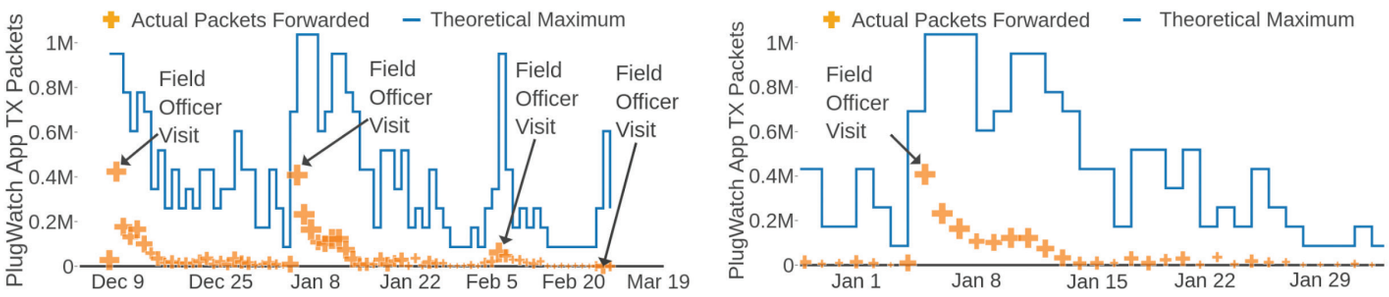
### Staying Alive

Android aggressively optimizes mobile phone behavior for user experience. We found that the OS would periodically close background services (which run even when an app is not displayed on screen) for garbage collection purposes. We used four techniques to circumvent the automatic garbage collection process:

1. The app generated notifications in the status bar, transforming the gateway service from a background service into a foreground service that was less likely to be killed by the OS under memory pressure.
2. The app forced the OS to keep the UI open on the screen, restarted the app on crash and boot, and disabled navigation buttons.

**WHEN TASKED TO RUN JUST A SINGLE APPLICATION CONTINUOUSLY WITHOUT ANY HUMAN INTERACTION, THE SMARTPHONE PLATFORM BECOMES SURPRISINGLY UNRELIABLE**

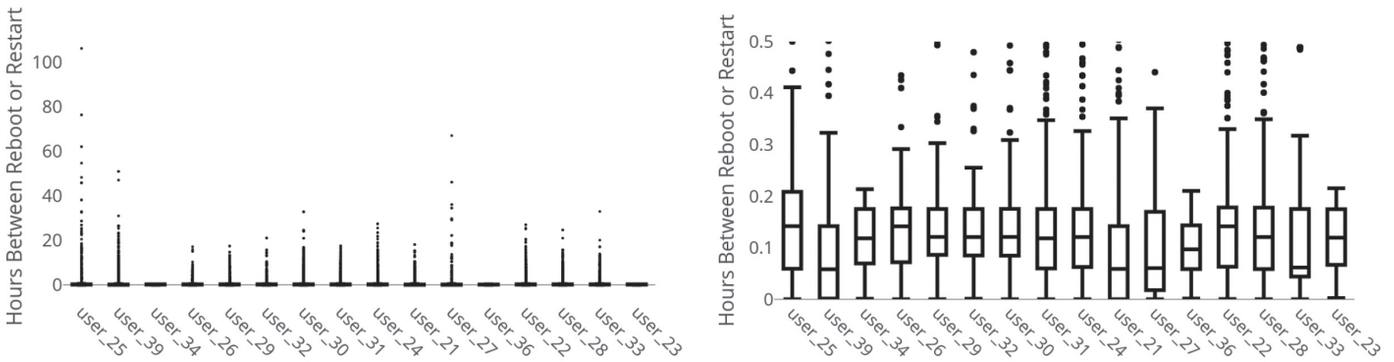
3. A separate watchdog process periodically woke and restarted the primary app process if it was no longer running, or rebooted the phone if a threshold number of restarts was exceeded. Likewise, the primary app process would restart this watchdog process if necessary.
4. The app used Android accessibility services to catch and automatically close all dialog boxes launched by the OS by using a heuristic of soft clicking either the only button or the leftmost button in the pop-up. This heuristic was sufficient to dismiss all system dialog boxes encountered during testing.




(a) Packets forwarded over entire deployment

(b) Close-up for month of January

**FIGURE 2.** Number of power measurement packets forwarded from the plug-load meter by the PlugWatch app to the cloud service. PlugWatch reports a power measurement at 1 Hz. Therefore, the daily theoretical maximum number of packets is calculated as number of seconds in a day multiplied by the number of phones reporting any packets during a day. After each field officer's visit to perform smartphone maintenance activities, the number of measurements received improved dramatically, then declined.




(a) Up time per-phone with all outliers

(b) Uptime per-phone with Y range near the upper quartile

**FIGURE 3.** Uptime in hours before an app restart or phone reboot. Box-and-whisker plots showing uptime over: (a) the full range of uptime; and (b) 0-0.5 hours. Each phone exhibits large variance in uptime, ranging from nearly zero to over 100 hours, which renders the boxes and whiskers nearly invisible in (a). However, (b) shows that the mean uptime for each phone was approximately 15 minutes or less. Unsupervised recovery from app crashes and phone reboots is therefore critical for gateway applications, but we found poor support for reliable, automatic, and interaction-free recovery in Android.

## DEPLOYMENT OUTCOMES

After deploying 16 PlugWatch systems in Zanzibar for four months, we found that PlugWatch underperformed in the field. As Figure 2 shows, even the two highest yield days of the deployment did not achieve half of the theoretical maximum number of packets. The plug-load meters were functional at the end of the deployment, leading us to conclude that most failures were due to the phones.

### Human Attention Helps

Field officers visited each household monthly to perform maintenance tasks including re-establishing the Bluetooth connection, installing available updates, clearing any system messages, and emptying the SD card. These visits account for the peaks in Figure 2. While our implementation efforts explicitly included routines that emulate human interaction, we could not match the performance boost of actual human attention.

### Long Tail of Errors

By far the most common error was the external watchdog app catching dead processes, which unfortunately provides no insight into why the core app failed. The next most common errors came from system service failures, which are bugs our app alone could not fix, leading in part to the need for frequent restarts. We observed a wide distribution of uptimes across the deployment as shown in Figure 3. The high variance suggests that several different factors shortened lifespan, making it hard to draw strong conclusions about the root cause of such frequent app restarts and phone reboots.

### Physical Catastrophes

By the end of the experiment, eight of the 16 deployed PlugWatch devices exhibited a visibly swollen battery, as Figure 4 shows. We hypothesize that battery problems led to the sharp drop-off in packets around February 5th, as seen in Figure 2. The worst battery was swollen to 270% its normal thickness. Six batteries were so swollen that they pushed the plastic phone backing out of its setting and the battery was no longer seated in the electrical contacts.

## LESSONS LEARNED

While our case study is a single deployment of a modest number of devices, it serves as a microcosm of a seemingly common deployment model: repurposing otherwise highly capable devices as an ad-hoc sensor data collection network.

### Human Interactions are Critical

Keeping humans in the loop during the experiment was ultimately necessary to keep the system operational. Unfortunately, our early design choices prevented us from taking advantage of this fact once discovered: we intentionally did not design a UI that would allow participants to easily interact with devices, did not include participant-device interaction in our IRB application, and screwed the phones into boxes with tamper-resistant hardware.


(a) Two swollen batteries


(b) Screen burn in

**FIGURE 4.** Observed hardware problems. (a) A battery and phone with its battery after deployment. Both demonstrate swelling. These issues did not manifest during thermal testing. (b) Screen burn-in from a device frozen in an invalid and unrecoverable state.

### User-Centric Phone Design Hurts Unattended Applications

Recent changes to Android have made things better for users but worse for unattended applications like PlugWatch.

Android 6.0 added runtime permissions, requiring users to manually approve each permission-protected function as it is invoked. Android 8 and 9 both constrain background operations, limiting the OS services that can wake an app and which sensors a background service can access. These changes prevent long-running background services from taking more than their fair share of resources, thereby improving user experience. However, these changes become significant obstacles when re-purposing phones as long-running gateways or sensor nodes without interactive users.

### Physical Environment Matters

Before deployment, we thermal tested the system at 120°F for 18 hours while logging battery temperature, battery capacity, and system reboot events, and found no unusual behavior. Despite this test, we were unable to predict the dangerous battery swelling that occurred in the heat and humidity of Zanzibar.

### You Cannot Future-Proof the Past

Some of the problems we experienced in Android 4.4.4 have been addressed in later versions of Android. For example, Android 5.0 added support for Google Enterprise Mobility Management (EMM) tools, which might have helped with provisioning phones, collecting debugging information, ensuring reliable OTA updates, and remotely accepting permissions. Android 8.0 introduced Google's Project Treble, which would have allowed us to replace or upgrade the buggy BLE driver on the PlugWatch phones without modifying the OS kernel itself. However, even if future versions of Android addressed all of the Android-related problems we encountered with PlugWatch, requiring more mature hardware or operating systems translates to abandoning billions of older phones already sold, as Figure 5 shows. This not only delays the viability of smartphone gateways until modern operating systems can penetrate the market, but abandons the most widely available computing resource across the globe. Increasing the reusability threshold, by even a minor version, could result in the consignment of millions of phones to landfills instead of productive second lives as gateways.

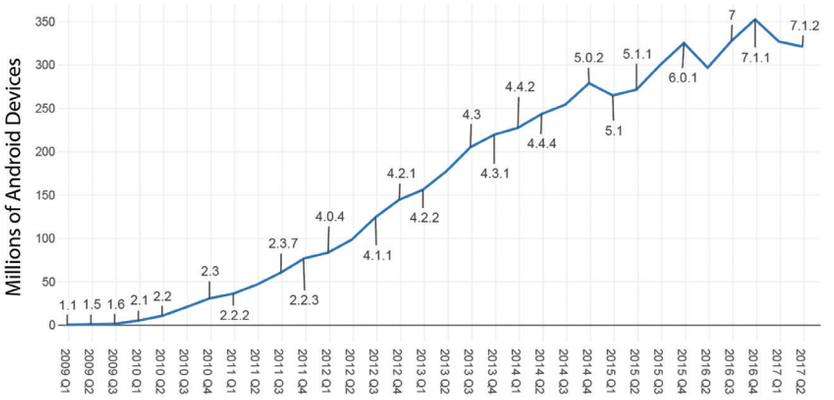



**FIGURE 5.** Global Android shipments from 2009 to 2017 and corresponding Android version number at time of shipment. Changing the supported version number by even a minor version number could potentially affect millions of phones.

## GOING FORWARD

Perhaps someday a user will be able to "donate their old phone's body to science" by triggering an OS upgrade to a gateway-friendly version of Android. For now, however, it is difficult to recommend repurposing Android smartphones as standalone gateways or sensors. After a year of sunk costs and lost time with Android, we gave up on smartphones and implemented nearly identical functionality with custom hardware in a matter of weeks. Our custom hardware has since spent nine months deployed in a similar environment with far better results. ■

**Noah Klugman** is a PhD student in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley, where he works on sensing electric power quality in areas that lack existing monitoring infrastructure, building systems to support sensor network deployments, and enabling societal scale participatory sensing.

**Pat Pannuto** is an NSF, NDSEG, and Qualcomm Innovation Fellow who is currently completing his PhD in the Department of Electrical Engineering and Computer Sciences at the University of California, Berkeley. His research is in the broad area of networked embedded systems, with contributions to computer architecture, wireless communications, mobile computing, operating systems, and development engineering.

**Matthew Podolsky** is a researcher at the University of California, Berkeley, where he is also the managing director of the Technology and Infrastructure for Emerging Regions and the associate director of Data Analytics at the Development Impact Lab. His research interests include information and communication technology for development, rural cellular communications, and energy measurement, monitoring, and reliability.

**Meghan Clark** is a PhD candidate at University of California, Berkeley, where she works with Prof. Prabal Dutta (UC Berkeley) and Prof. Mark W. Newman (University of Michigan) on improving technologically mediated interactions with the built environment. Her research draws upon many disciplines, including systems, natural language processing, and mixed reality.

**Jay Taneja** is an assistant professor of Electrical and Computer Engineering at the University of Massachusetts, Amherst. His research studies the application of computing tools for measuring infrastructure in developing regions. He formerly led the Energy research team at IBM Research, Africa. He received a PhD in Computer Science from the University of California, Berkeley.

**Prabal Dutta** is an associate professor of Electrical Engineering and Computer Sciences at University of California, Berkeley. His research interests straddle the hardware/software interface and include wireless, embedded, networked, and cyber-physical systems. He received a PhD in Computer Science from the University of California, Berkeley.