Exploring personally relevant making

Tiffany Chao and Peter Enns

University of Maryland, College Park
{tjyc, slunk}@umd.edu

1 Introduction

The ability to create something usable, something personally relevant, is immensely powerful. And with increasingly accessible parts and processes, this has become far more attainable for electronic devices.

Inspired in part by our time in Tangible Interactive Computing [4], we wanted to explore something that was meaningful for both of us. We chose coffee: more specifically, how to not burn ourselves while drinking it.

2 Ideation

We were interested, more generally, in what connected devices could afford: smart "forget-me-nots", perhaps? New ways to keep track of small children? Most of our initial thoughts were too broad or too vague, and ultimately discarded. What we converged on was something smaller, but still compelling: a coffee cup.

There's definitely a niche for a smart cup: see [6], for example, or [7]. Cups are ubiquitous already—imagine if you could successfully change them? Starbucks, for instance, hopes to serve 5% of its store-made beverages[1] in personal tumblers by 2015 [5]. Perhaps additional features would make reusable tumblers more attractive?

That said, our goals were far humbler. We just wanted to improve our personal coffee-drinking experience. Drink temperature stood out; the rest developed organically from there.

3 Picking parts

Our base requirements were relatively simple:

1. a means to measure temperature, and
2. a means to convey that temperature to a smartphone.

Our priorities were accordingly straightforward:

1. prototyping ease (over production readiness), and

[1] For a sense of scale: 1.8% in 2013 meant 46.9 million beverages and over 1.4 million pounds of paper [5].

2. functionality (over form).

Further trading form for reliability, we settled on an Arduino Uno (for some sensor-side computing power), a RedBearLab BLE shield (for Arduino-smartphone communication), and a DS18B20 sensor (for reasonably precise temperature readings).[2]

4 The prototype

4.1 Major components

Arduino Uno The Arduino Uno [2] is a microcontroller board with an 8-bit Atmel AVR microcontroller at its core. It has 14 digital input/output pins, 6 analog inputs, and a standard USB connection for easy programming and serial communication. Importantly, its standardized pin connectors support a wide variety of modular "shields", offering vast extensibility.

RedBearLab BLE shield RedBearLab's Bluetooth Low Energy (BLE) shield [3], based on the Nordic nRF8001, is a perfect example of how the Arduino Uno can be trivially extended. Stacking the shield onto a compatible board (like the Uno) lets the board function as a BLE peripheral, enabling two-way communication between it and nearby BLE centrals.

Android smartphone The Android operating system (OS) is Google's open source mobile OS. Android applications are developed in Java using the Android software development kit (SDK). With extensive online documentation and useful tools (debugger, libraries, etc.) freely available, the barrier to entry is relatively low. And since Android applications have access to abstract APIs that interface with a wide range of hardware typically found on smartphones (e.g. Bluetooth modules), they're quite capable of interacting with other devices in interesting ways. We developed with two Android smartphones: a Samsung Galaxy S4 mini and an LG Nexus 4.

4.2 Communication protocol

The Uno is the peripheral; the smartphone is the central. Peripheral-to-central messages map sensor names to their latest readings, and are sent in JSON (see Figure 1). Central-to-peripheral messages describe per-sensor threshold adjustments, and are sent in plain text (see Figure 2).

The typical use case has the Uno sending sensor readings (raw data) and receiving per-sensor threshold adjustment requests (vaguely resembling remote procedure calls). The latter is far less common.

[2] We already had two Android smartphones (a Samsung Galaxy S4 mini and an LG Nexus 4).

{"temp":25,"press":1}

Fig. 1. Example Uno-to-phone message

set temp lo 20

Fig. 2. Example phone-to-Uno message

4.3 Current features

Drink temperature notifications Baristas don't always get your name right (though they do try); and even if they do, they may mispronounce it. Then once you've retrieved your coffee, the next phase begins: when can I actually drink this?

We tackle these two problems with three notifications: one when a hot liquid is poured into the cup (so you know when your drink is about done), one when the liquid reaches a reasonable temperature (so you know when it's safe to drink), and one when the drink has become too cold (so you can perhaps ask for a refill).[3]

"Forget-me-not" notifications Keys, wallets, phones, bags—it's incredibly easy to misplace *something*. And since a "forget-me-not" feature would be useful for many potential peripherals, we decided to add one.

Any connected peripheral can be tagged as something that shouldn't be forgotten. When the connection between phone and peripheral is lost, the user receives the appropriate reminder. While this is admittedly a simplistic mechanism, it's proven to be a reasonable approximation.[4]

5 Conclusion

We may yet explore additional sensors (an accelerometer to detect when the cup's been knocked over, for example), or augment the interaction with inferences based on data already available Android-side. But perhaps it would be more interesting to consider other prototyping techniques altogether. If we used conductive tape instead of wires and breadboards, would we have come up with something different? If we'd chosen a much smaller peripheral, would we have pursued different interactions? Making is very much a process of bricolage [11]; efforts to expand related prototype techniques are ongoing.[5] If we revisited the

[3] The high and low temperature thresholds actually allow us to address cold drinks as well. The states there become: (i) not empty, (ii) cold enough, and (iii) too warm.

[4] We'll experiment with heartbeat messages in the future.

[5] And quite varied: recent examples include a 3D printer that uses wool yarn [9], low-fidelity wireframes printed directly in 3D space [12], inkjet-printed circuits [1, 10], modular circuit stickers [8], and homemade conductive and insulating doughs [13].

same problem a year from now, would we have produced something entirely different? Maybe. Maybe it won't even be a problem anymore. We look forward to new developments, all the same.

References

1. AgIC. http://agic.cc/.
2. Arduino Uno. http://arduino.cc/en/Main/arduinoBoardUno.
3. BLE Shield. http://redbearlab.com/bleshield/.
4. CMSC838f: Tangible Interactive Computing. http://cmsc838f-s14.wikispaces.com/.
5. Cups and Materials. http://www.starbucks.com/responsibility/environment/cups-and-materials.
6. Vessyl. https://www.myvessyl.com/.
7. Hiroshi Chigira, Masayuki Ihara, Minoru Kobayashi, Akimichi Tanaka, and Tomohiro Tanaka. Heart rate monitoring through the surface of a drinkware. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 685–689, New York, NY, USA, 2014. ACM.
8. Steve Hodges, Nicolas Villar, Nicholas Chen, Tushar Chugh, Jie Qi, Diana Nowacka, and Yoshihiro Kawahara. Circuit stickers: Peel-and-stick construction of interactive electronic prototypes. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 1743–1746, New York, NY, USA, 2014. ACM.
9. Scott E. Hudson. Printing teddy bears: A technique for 3d printing of soft interactive objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 459–468, New York, NY, USA, 2014. ACM.
10. Yoshihiro Kawahara, Steve Hodges, Benjamin S. Cook, Cheng Zhang, and Gregory D. Abowd. Instant inkjet circuits: Lab-based inkjet printing to support rapid prototyping of ubicomp devices. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 363–372, New York, NY, USA, 2013. ACM.
11. David A. Mellis and Leah Buechley. Do-it-yourself cellphones: An investigation into the possibilities and limits of high-tech diy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 1723–1732, New York, NY, USA, 2014. ACM.
12. Stefanie Mueller, Sangha Im, Serafima Gurevich, Alexander Teibrich, Lisa Pfisterer, François Guimbretière, and Patrick Baudisch. Wireprint: 3d printed previews for fast prototyping. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 273–280, New York, NY, USA, 2014. ACM.
13. Matthew Schmidtbauer, Samuel Johnson, Jeffrey Jalkio, and AnnMarie Thomas. Squishy circuits as a tangible interface. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 2111–2116, New York, NY, USA, 2012. ACM.