

One-Button Recognizer: Exploiting Button Pressing Behavior for User Differentiation

Henning Pohl, Markus Krause, Michael Rohs

FG Mensch-Computer Interaktion

University of Hannover

Hannover, Germany

{henning.pohl, markus.krause, michael.rohs}@hci.uni-hannover.de

ABSTRACT

We present a novel way to recognize users by the way they press a button. Our approach allows low-effort and fast interaction without the need for augmenting the user or controlling the environment. It eschews privacy concerns of methods such as fingerprint scanning. Button pressing behavior is sufficiently discriminative to allow distinguishing users within small groups. This approach combines recognition and action in a single step, e.g., getting and tallying a coffee can be done with one button press. We deployed our system for 5 users over a period of 4 weeks and achieved recognition rates of 95% in the last week. We also ran a larger scale but short-term evaluation to investigate effects of group size and found that our method degrades gracefully for larger groups.

Author Keywords

User recognition; physical interaction; sensing; lightweight interaction; adaptive user interfaces; context recognition.

ACM Classification Keywords

H.5.2 [Information Interfaces and Presentation]: User Interfaces—*Input devices and strategies*

INTRODUCTION

Differentiating who is using a device among a group of co-operating users is a frequent task in ubiquitous computing systems. We might, e.g., be interested in who just entered a room, who picked up a phone, or who is activating the dish washer. In such scenarios, security might be less important than convenience, i.e., minimizing the burden on users. Such convenience can be achieved by using biometrics or instrumenting users (e.g., RFID tags). However, this comes at a cost of privacy or can be impractical or out of place in some contexts (e.g., at home).



Figure 1. Pushing the button next to the coffee machine, William is recognized and his coffee tally is updated.

A number of ways to recognize users with low effort have been introduced [14, 19, 20]. Along the same lines, we propose differentiating users just by how they press a physical button. Because an explicit action is required, this avoids problems of accidental activations as in more automated systems. Furthermore, this is a common action—if every button in an environment has the capability to recognize who pressed it, an abundance of context information is generated. Moreover, pressing a button is a low-focus activity and can be done casually while being engaged elsewhere [18], building on gross motor skills, inexact, and inattentive interaction [5, 8].

User recognition via button pressing is particularly suitable in one-off interactions, where users are not engaged for long and where, at the extreme, the interaction itself can be reduced to one button press. One such scenario, that we take as an example here, is tallying coffee consumption in an office (see Figure 1). Often this is done using a paper tally sheet where users mark every coffee. However, this requires tedious manual balancing at some point and requires cooperating users, as tally marks can easily be manipulated. We propose changing the task from making a mark on a paper sheet to pressing a button next to the coffee machine—and eventually replacing the button on the machine with a button that recognizes users.

We built two prototypes of buttons that can recognize who pressed them: (1) a portable push button that integrates a pressure and a distance sensor, and (2) a wall-mounted version that extends a standard light-switch with a distance sensor. These sensors are low cost, compact, and robust to changing conditions. We did not use computer vision methods to make the approach resilient against lighting changes. This choice of sensors also allows for lightweight and transient interactions. Prolonged contact of the user with the interface is not required. The two prototypes are similar to the button designed by Spelmezan et al. [23], whose focus is on recognizing explicit gestures, rather than implicitly recognizing users.

In two evaluations, we investigate (1) recognition rates for different group sizes from a large set of 39 users, and (2) in-use system behavior over a 4 week period of time with a fixed set of 5 users. In our long term deployment, we observed fast improvement over time with over 94 % recognition rate in the last week. Users can correct mis-classifications with another button press—a low cost disambiguation approach. This extra action boosts recognition rates to 99 %.

RELATED WORK

There have been a number of approaches to low-cost user recognition. Commonly, biometric features such as fingerprints [7] or hand contours [21] are used. However, biometric data can be privacy sensitive and users might have objections to providing it for a task such as getting a coffee. Users' typing patterns also fall under this category and have been extensively exploited [11–15, 17]. Typing approaches, however, are not applicable to short, transient interactions, where no text entry is required. Computer vision has been used to identify users based on their shoes [20], backs of hands [19], or in-air gestures [1]. Any method using vision relies on proper lighting though, restricting possible scenarios. Other properties used include walking behavior [24], body capacitance [6], deviations in touch interaction [4], or combinations of many sensors for passive user identification [22].

ONE-BUTTON RECOGNIZER

Our recognizer builds on the observation that button-pressing behaviors differ slightly. This was previously illustrated by Kim et al. [10], who used such pressure profiles to recreate haptic sensations for augmented reality buttons. Instead of recreating the button pressing sensation, we set out to use the nuances in button pressing for differentiating users. As shown in Figure 2, the sensor readings of a button press vary distinctively between different users.

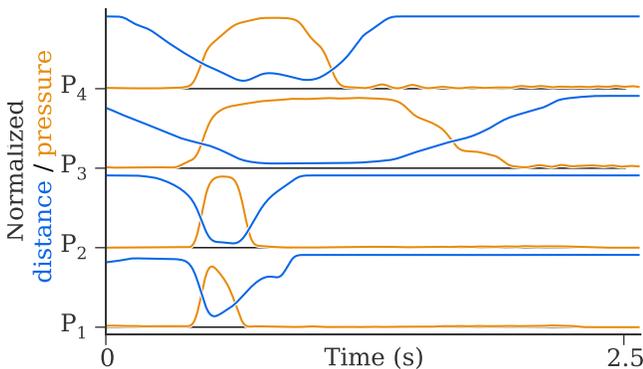


Figure 2. Comparing the normalized distance and pressure curves for 4 participants. The variation in button pressing behavior is apparent.

Button Prototypes

We designed two prototypes for two distinct form factors: (1) desk, and (2) wall-mounted. Both buttons include a *Sharp GP2Y0A41SK0F* infrared distance sensor (4–30 cm operating range) to detect how a hand approaches the button. The sensor works reliably over a wide range of lighting conditions. Both prototypes take pressure (where applicable) and distance readings via the 10 bit ADC of an *Arduino Nano*. Data is sampled at about 400 Hz and, via a serial connection, relayed to a host computer for further processing.

Desk Button

The desk button is 3D-printed in multiple parts (shown in Figure 3). Four springs separate the top of the button from the base to provide return force and give the button a proper feel. Force sensing is performed with an *Interlink FSR402* pressure sensor (sensitive in the 0.2–20 N range), attached below the spring contact plate. A small protrusion on the bottom of the contact plate focuses the force on the pressure sensor. The sensor gives us a pressure profile over the whole duration of a button press. However, we threshold the pressure values to derive a binary switch state. In this prototype the distance sensor is placed in front of the button.

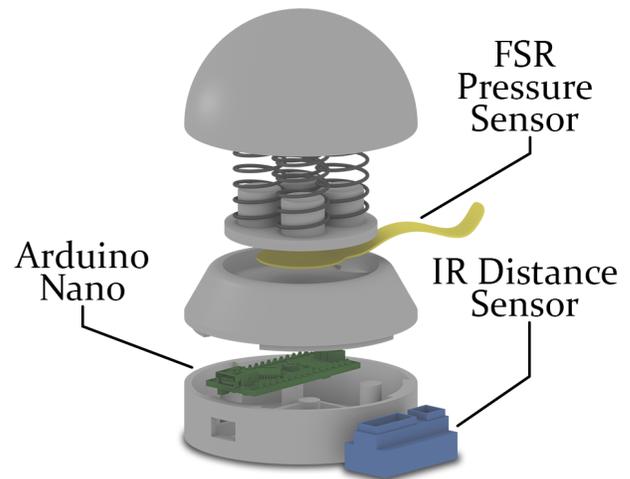


Figure 3. Our pressure-sensitive button contains an FSR sensor in a 3D-printed enclosure. Springs provide return force and push resistance. An IR distance sensor is attached at the side.

Wall-Mounted Button

For the wall-mounted prototype, we modified a *Voltomat MIKRO* push-button light switch. This is a standard product from a hardware store. We mounted this button on a wall and embedded the distance sensor underneath (see Figure 4). Future versions could integrate the distance sensor in the button (there is some space for a light which could be repurposed). Due to space constraints, this button did not integrate a pressure sensor. Instead, the switch itself is connected to the *Arduino Nano* which reads and relays the binary switch state.

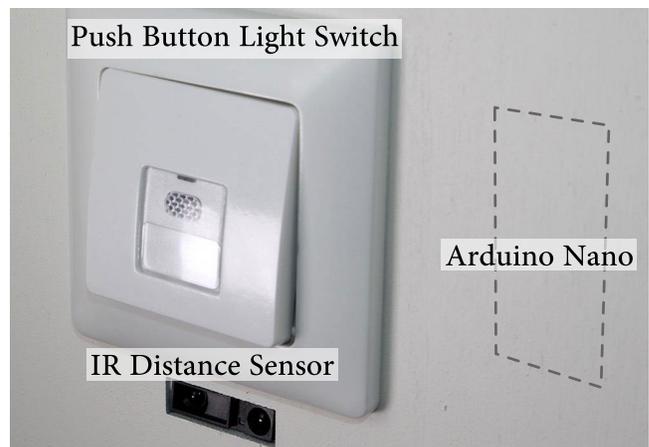


Figure 4. We augment a standard light switch with a distance sensor to enable identification of who pressed it. Future switches could embed the sensor directly in the enclosure.

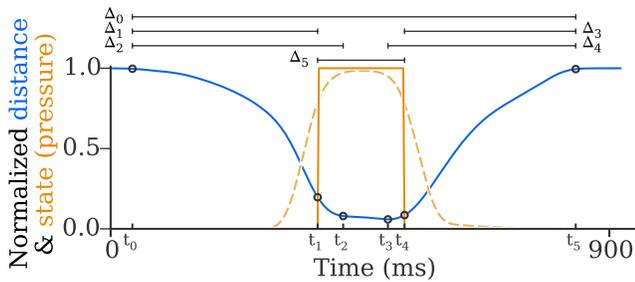


Figure 5. We build features from six times along a button press trajectory: attack start t_0 , press start t_1 , attack end t_2 , release start t_3 , press end t_4 , and release end t_5 . Features are then defined as time differences for six distinct phases of the button press. For the desk button, we threshold the pressure values (shown dashed) to compute button state.

Feature Extraction

We derive our features from six points along the trajectory of a button press (see Figure 5). Points t_1 and t_4 mark the beginning and end of the actual button press. In the wall button this is just the switch state. For the desk button, however, we threshold the continuous pressure values at 80% of the maximum to derive binary button state.

Starting from those two points, we search for the start and end of the attack (t_0 and t_2) and release phase (t_3 and t_5) respectively. Those points are derived from the local extrema of the distance sensor signal around the start/end of a button press. Note that those points can be on either side of the button hit event, depending on the way the button is pressed.

From those points we derive six features. The overall interaction time Δ_0 and button down time Δ_5 capture speed. We also take into account two different measures of attack and release timing. For attack timing, we include the time it takes from the start of the approach to both (1) the button pressing Δ_1 , and (2) the end of the approach Δ_2 . Similarly, during release we use overall withdrawal time Δ_4 and the time from the release of the button Δ_3 . We use those features to discriminate two different button pressing behaviors: (1) Some users first press the button, then lower their palm. In this case Δ_1 is longer than Δ_2 . (2) Other users attack the button with their palm already lowered. In this case Δ_2 is longer than Δ_1 . This behavior holds inversely during the release phase.

Training the Recognizer

We use a random forest classifier, because they directly handle multiple classes, and are less sensitive to outliers [3]. In our evaluation, we only used a fixed numbers of training samples per user. Our classifier generated 10 random trees per forest using Gini impurity [2] as split criterion.

EVALUATING RECOGNITION PERFORMANCE

In a first study, we investigate how well we can distinguish users in groups of different size and how many button presses would be needed for training. For this, we set up our desk prototype for two weeks in our lab (for use by staff and students) and in a classroom (for use of students in a course). Participants each time entered their name on the connected laptop and then pressed the button. We did not instruct participants in any way as to how to press the button. Overall, we gathered data from 44 participants (7 female) in this way. However, in our analysis we remove all users who pressed the button less than 30 times (leaving us with 39 users).

We investigate the performance for group sizes of 2 to 10 users and with 2 to 30 button presses used as training data (at intervals of 2). For every combination of group size and button presses, we generate 1000 permutations from our data for subsequent testing. We do this to test the robustness of our classifier against group effects, such as subsets of users with similar button pressing behavior.

Accuracy testing for each permutation is done with leave-one-out sampling [9]. This yields binary results, which we aggregate to a mean accuracy score per permutation. Testing one sample against a classifier trained on all remaining data fits well with our intended use case: classifying one new button press at a time. For each combination, we aggregate these accuracies and calculate the 95% confidence interval of the mean via bootstrapping with 10000 iterations.

Results

How well we can identify users largely depends on the size of the user set. In Figure 6—showing recognition rate for all tested combinations—this can be seen in the vertical orientation of the contour lines once a minimum amount of training data is used. When using 24 button presses for training, e.g., accuracy drops almost linearly from 95.0% (two users) to 78.2% (10 users). Overall, accuracy is less sensitive to the amount of training data. For groups of 5 users, e.g., recognition rates exhibit a quick rise from 69.4% for 2 samples per user to 82.3% for 10 samples. This is followed by slight improvement with 30 samples giving 90.3% recognition rate.

We saw high variance in accuracy when using small numbers of training samples. For example, using only 2 button presses is quite unstable and we observed $\pm 16\%$ standard deviation of bootstrapped recognition rates. The stability of the classifier goes up for larger training sets, e.g., being as low as $\approx \pm 3\%$ for groups of ten users when using more than 10 samples.

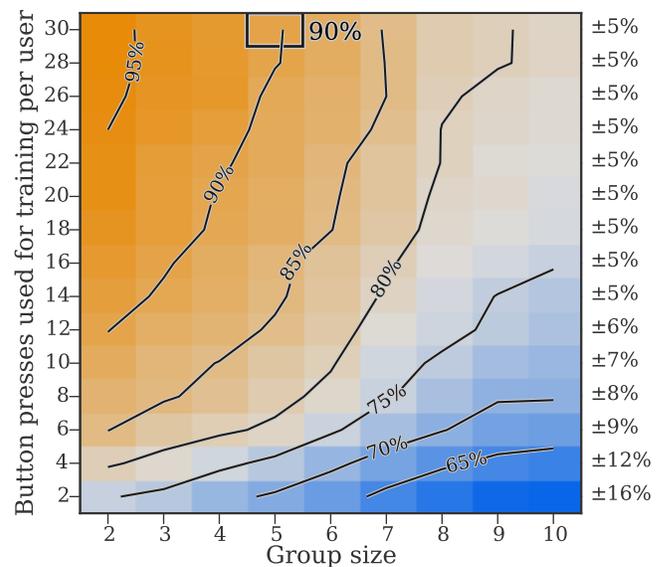


Figure 6. Recognition rates for group sizes from 2 to 10 users and button press data from 2 to 30 presses. Recognition rates were computed using 10000 bootstrap samples. On the right we show the average standard deviation for each for row—training a recognizer with low numbers of button presses results in high variance, which goes down with more training data. Group size does not influence variability as much.

From this data we can compute the expected number of button presses during an interaction needed to reliably identify a user. Table 1 shows this number for different group sizes. For groups of 5 users with 30 samples each, a user is recognized at the first button press with 90.3% probability and at the second button press with 99.1% probability. After the button has been pressed three times, the expected probability of recognition is over 99.9%. Given the very low effort to correct a mistake (average time of a button press is $M = 0.4s$, $SD = 0.1s$), a possible additional button press takes but a mere instant.

Group size	2	3	4	5	6	7	8
$E[\# \text{ of presses}]$	1.04	1.08	1.11	1.13	1.18	1.20	1.23
$E[\delta \text{ (1st press)}]$	0.04	0.08	0.13	0.14	0.15	0.17	0.19
$E[\delta \text{ (2nd press)}]$	-	0.00	0.01	0.02	0.02	0.03	0.04
$E[\delta \text{ (3rd press)}]$	-	-	0.00	0.00	0.00	0.01	0.01

Table 1. The expected number of required presses and error rate δ go up with group size. It is unlikely to require more than 2 presses.

EVALUATING PROLONGED USAGE

In a second study we set out to evaluate an actual deployment of the button. We installed the wall-mounted prototype in our lab (hung in an office as shown in Figure 4) and provided it the names for a group of five users from our lab. Over a period of 4 weeks, those users would pass by to press the button every time they got a coffee.

While users in the first study did provide their name before a button press, this step was left out in this setup. Instead of providing ground truth before an interaction, we introduced a disambiguation mechanism in the second study to allow users a way to correct a wrong recognition. After a user pressed the button, audio feedback is provided via a text-to-speech engine (“Thanks <Name>”). If a user is not properly recognized, she can press the button again within the next 3 seconds to override that last recognition. Thus ground truth is provided by the final *user-accepted recognition*. We allowed cancellation by pressing the button down for at least 3 seconds, however this was not used at all during the deployment.

At first the button is untrained, i.e., has no recorded samples for any user. Thus, initially, performance is poor while the system is collecting samples (and users have to make use of the disambiguation mechanism by pressing the button again). We limit the number of training samples to 40 per user. This provides sufficient data for recognition but also allows the system to adapt to changing button pressing behavior.

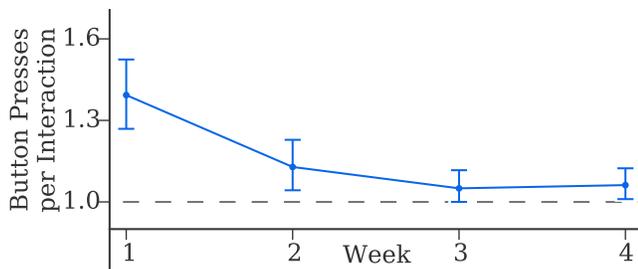


Figure 7. Over the course of 4 weeks, the number of times users had to press the button for one interaction (for disambiguation) goes down and approaches the theoretical minimum of 1. Error bars show 95% confidence intervals.

Results

Overall, we logged 372 interactions with the button. We can see in Figure 7 that the number of required button presses goes down over time. During the first week, the system is still learning and users are getting used to it. However, even in that first week, users on average only need to disambiguate every third button press. Over the following weeks, the recognition rate improves and approaches the theoretical minimum of 1.

Overall, we were very pleased with the performance of the button in our lab setting. In the beginning the system was still adapting though, but quickly reached acceptable recognition levels. A confusion matrix for the last two weeks is shown in Table 2. With low instances of required disambiguation, using the button was mostly as easy as walking by, pressing it, and walking away.

Actual \ Predicted	User 1	User 2	User 3	User 4	User 5
User 1	95.0%	0.0%	0.0%	5.0%	0.0%
User 2	0.0%	100.0%	0.0%	0.0%	0.0%
User 3	0.0%	7.1%	92.9%	0.0%	0.0%
User 4	6.2%	0.0%	3.1%	90.6%	0.0%
User 5	2.8%	0.0%	0.0%	8.3%	88.9%

Table 2. Confusion matrix for last two weeks of the deployment test.

CONCLUSION

We proposed a low-effort user recognition approach that piggybacks on button presses. It links identity information to button presses and thus integrates recognition and action. The employed sensors and recognition system are simple enough to be integrated in commodity buttons, essentially disappearing [25]. Buttons enhanced in this way can serve sources of context information (as in [16]). The evaluations show that the prototypes are reliable enough to differentiate between users in small groups, such as in offices. The approach minimizes the burden on users.

The button-recognizer concept addresses small cooperating groups of people who share a resource. Labs, e.g., often have shared computers for devices such as 3D printers, which could boot and load user profiles automatically when the power button is pressed. Similarly, families’ home entertainment systems could switch profiles and play one’s favorite music with a single button press.

Large buttons, like conventional light switches, enable quick, inexact, and inattentive action. They do not require fine motor skills. Our approach maintains these properties. However, it also gives users the opportunity to evolve their behavior. Users may deliberately modify their button-pressing behavior to communicate additional information or to become more distinct from others and thus easier to differentiate. Users could even “teach” their button-pressing behavior to others (akin to a secret handshake), allowing trusted users to impersonate them.

As for future work it is conceivable to extend this approach to other objects that users manipulate in their everyday environment, such as door handles, window handles, door hinges, cabinets, etc. User identification for actions on these objects enables a wide range of application possibilities.

REFERENCES

1. Aumi, T., and Kratz, S. AirAuth: Evaluating In-Air Hand Gestures for Authentication. *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 2014.
2. Breiman, L. Technical Note: Some Properties of Splitting Criteria. *Machine Learning*, 24(1), July 1996: 41–47.
3. Cieslak, D. A., and Chawla, N. V. Learning Decision Trees for Unbalanced Data. *Proc. PKKD*. 2008, 241–256.
4. De Luca, A., Hang, A., Brudy, F., Lindner, C., and Hussmann, H. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. 2012, 987–996.
5. Dix, A. Beyond intention - pushing boundaries with incidental interaction. *Proceedings of Building Bridges: Interdisciplinary Context-Sensitive Computing*. 2002, 1–6.
6. Harrison, C., Sato, M., and Poupyrev, I. Capacitive Fingerprinting: Exploring User Differentiation by Sensing Electrical Properties of the Human Body. *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*. 2012, 537–544.
7. Holz, C., and Baudisch, P. Fiberio: A Touchscreen That Senses Fingerprints. *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST '13*. 2013, 41–50.
8. Hudson, S. E., Harrison, C., Harrison, B. L., and LaMarca, A. Whack Gestures: Inexact and Inattentive Interaction with Mobile Devices. *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction - TEI '10*. 2010, 109–112.
9. Kearns, M., and Ron, D. Algorithmic Stability and Sanity-Check Bounds for Leave-One-Out Cross-Validation. *Neural Computation*, 11(6), Aug. 1999: 1427–1453.
10. Kim, Y., Kim, S., Ha, T., Oakley, I., and Woo, W. Air-Jet Button Effects in AR. *Proc. Lecture Notes in Computer Science*, 384–391.
11. Krause, M. A Behavioral Biometrics Based Authentication Method for MOOCs that is Robust Against Imitation Attempts. *Proceedings of the first ACM Conference on Learning@Scale*. 2014, 201–202.
12. Leggett, J., Williams, G., Usnick, M., and Longnecker, M. Dynamic identity verification via keystroke characteristics. *International Journal of Man-Machine Studies*, 35(6), Dec. 1991: 859–870.
13. Mock, P., Edelmann, J., Schilling, A., and Rosenstiel, W. User Identification Using Raw Sensor Data From Typing on Interactive Displays. *Proceedings of the 19th international conference on Intelligent User Interfaces - IUI '14*. 2014, 67–72.
14. Monrose, F., and Rubin, A. D. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 16(4), Feb. 2000: 351–359.
15. Ogihara, A., Matsumura, H., and Shiozaki, A. Biometric Verification Using Keystroke Motion and Key Press Timing for ATM User Authentication. *2006 International Symposium on Intelligent Signal Processing and Communications*. Dec. 2006, 223–226.
16. Paradiso, J. A., and Feldmeier, M. A Compact, Wireless, Self-Powered Pushbutton Controller. *Proceedings of the 3rd International Conference on Ubiquitous Computing - Ubicomp '01*. 2001, 399–304.
17. Peacock, A., and Wilkerson, M. Typing patterns: a key to user identification. *IEEE Security & Privacy Magazine*, 2(5), Sept. 2004: 40–47.
18. Pohl, H., and Murray-Smith, R. Focused and Casual Interactions: Allowing Users to Vary Their Level of Engagement. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. 2013, 2223–2232.
19. Ramakers, R., Vanacken, D., Luyten, K., Coninx, K., and Schöning, J. Carpus: A Non-Intrusive User Identification Technique for Interactive Surfaces. *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*. 2012, 35–44.
20. Richter, S., Holz, C., and Baudisch, P. Bootstrapper: Recognizing Tabletop Users by Their Shoes. *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. 2012, 1249–1252.
21. Schmidt, D., Chong, M. K., and Gellersen, H. Hands-Down: Hand-Contour-Based User Identification for Interactive Surfaces. *Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries - NordiCHI '10*. 2010, 432–441.
22. Shi, W., and Yang, J. SenGuard: Passive user identification on smartphones using multiple sensors. *2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications - WiMob '11*. Oct. 2011, 141–148.
23. Spelmezan, D., Appert, C., Chapuis, O., and Pietriga, E. Controlling Widgets With One Power-Up Button. *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST '13*. 2013, 71–74.
24. Suutala, J., and Rönning, J. Methods for person identification on a pressure-sensitive floor: Experiments with multiple classifiers and reject option. *Information Fusion*, 9(1), Jan. 2008: 21–40.
25. Weiser, M., and Brown, J. S. The Coming Age of Calm Technology. In: *Beyond Calculation: The Next Fifty Years of Computing*. Vol. 01. July. New York, NY, USA: Copernicus, 1997, 75–85. ISBN: 0-38794932-1.