Evaluation of a Software Architecture Supporting Android Applications for Users with Motor Disabilities

Olibário José Machado-Neto
University of São Paulo
olibario@icmc.usp.br

Maria da Graça Campos Pimentel
University of São Paulo
mgp@icmc.usp.br

ABSTRACT

A software architecture is an organization that contains the structure of a software and the relationship of its components. As a result, software architectures dictate the way software is created and updated. We propose a software architecture to help developers to create Android-based applications for users with motor disabilities – specifically. The architecture supports using one or more hardware components of mobile devices including built-in sensors, camera and microphone. Also, it encompasses a straightforward way of using and integrating such resources, which may lead to applications that provide alternative ways for accessing and managing data by users with disabilities. The solution also provides functionalities to work with raw sensor data, and offers a model for storing medical information of users. An evaluation with 19 software developers indicate that the architecture can be useful for creating not only solutions for people with motor disabilities, but diverse applications.

KEYWORDS

Mobile Computing, Accessibility, Assistive Technology

1 INTRODUCTION

Interacting with touch-based mobile devices is more natural than interacting with traditional personal computers mainly because of movements such as pinching and sliding the finger, which are easier to be understood and to be performed than interacting with the mouse [8]. Moreover, the built-in hardware of such devices can also be used to facilitate the interaction with the user.

Despite more natural, we noticed that the literature lacks a software architecture to facilitate the development of applications for people with disabilities by using and integrating the hardware of mobile devices. In this study, we present an architecture to help developers to create such applications in less time and more easily. With such architecture, we also intent to make it easier to implement new features and to update existing software functionalities. In mobile development, all sensors and hardwares can be used simultaneously, the usage of these components is common, so facilitating their usage and integration is a positive feature for developers.

In:XVII Workshop de Teses de Dissertações (WTD 2017), Gramado, Brasil. Anais do XXIII Simpósio Brasileiro de Sistemas Multimídia e Web: Workshops e Pôsteres. Porto Alegre: Sociedade Brasileira de Computação, 2017.

ISBN 978-85-7669-380-2.

Although our software structure might be suitable for any operating system of mobile devices, we implemented the infrastructure in the Java programming language, for Android. The differences between our architecture and the ones we found in the literature are: it is intended for the mobile context; it can be used to record medical data; it integrates the built-in hardware of mobile devices in a straightforward way in order to create applications that be used for assistive purposes.

We present an overview of the architecture and the result of an evaluations carried out with 19 Android developers who used the architecture for four hours to implement one application to take pictures that uses both the camera, the vibrator and the gyroscope.

In this paper, we summarize related work in Section 2, we outline the methodology we used in Section 3, we overview our proposed architecture and discuss its evaluation by software developers in Section 4, and make our final remarks in Section 5.

2 RELATED WORKS

The use of software architecture to support accessibility is well documented. *WebBee* [12] is an architecture that executes scripts to access web contents that are used to create accessible mobile applications. Zacarias et al. [13] proposed an architecture to support question-answering applications, which are representative nowadays, and can eventually be used for accessibility purposes. Baguma and Lubega [2] proposed an architecture for developing web applications for people with disabilities, that contains guidelines that can be used by developers in order to obtain accessible web applications. Despite their practical relevance, none of these studies focus on the creation of applications for mobile devices.

Senatore et al. [11] developed an infrastructure to support people with motor disabilities in doing daily tasks. Alharthi et al. [1] proposed an architecture that relies on context awareness to adapt e-health applications according to the needs of users with disabilities. However, these studies did not use mobile computing.

Some studies that explicitly uses resources of mobile devices have also been reported, but software architecture is not their focus. The *i-Sleep* [6], for example, is a system that monitors an individual's sleep quality using off-the-shelf smartphone, by using the built-in microphone to detect events related to sleeping, such as body movement, couching and snoring. A similar work has been reported by Chen et al. [3], who used sensor-based inference algorithms that are executed in order to predict sleep duration of users. *RunBuddy* [7] is a smartphone system for monitoring the running rhythm of the user, by means of the accelerometer data and the sound of the human breath. Feijó Filho et al. [5] used the pressure sensor of mobile devices to propose solutions that substitute conventional basic telephone function by breath commands, such as puffing. Ohnishi et al. [9] proposed a system that used the camera to identify characters of the surrounding environment, and dictate them via voiceover for blind users. Façanha et al. [4] developed a mobile application that provides a writing tool based on braille, which also gives audio feedback for people with visual disability. Oliveira et al. [10] created a mobile application to teach children with learning impairments via gesture inputs. All of these studies report solutions that rely on one or more built-in hardware of mobile devices in order to provide accessibility. This is the motivation for us to propose a software architecture that provides data and parameters of such hardware for software developers. The data can be further used to record medical data and sent to a centralized server, which are also features supported by the architecture.

3 METHODOLOGY

Literature Review. First, we performed two extensive reviews of the literature regarding assistive technology development and software architectures with purposes similar to ours. The first review consisted of reading the abstracts of all articles published, from 2005 to 2015, in the ACM SIGACCESS Conference on Computers and Accessibility (ASSETS) and the ACM Transactions on Accessible Computing (TACCESS), two important sources of accessibility research reports. We categorized each of these articles according to the interaction modality of the solutions (visual, haptic or auditory), the devices used (camera, microphone, personal computer, smartphone, tablet, among others) and resources used (sensor, camera, microphone, vibrator, mouse, keyboard, among others). We read all articles related to mobile devices.

The second literature review was based on the results from searches of articles in English and in Portuguese in ACM Digital Library[1], IEEExplore[2] and ScienceDirect[3]. We adapted the syntax of our search string to suit the search requirements of each digital library. Overall, the string we used was: *(Title:((architecture OR architectures OR infrastructure OR infrastructures OR "design pattern" OR "design patterns" OR framework OR frameworks) AND (assistive OR accessibility OR accessible OR disability OR disabilities OR impairment OR impairments OR disabled OR impaired))).*

We found 38 results in ACM, 81 in IEEExplore and 42 in ScienceDirect. After importing the results to Mendeley excluding repeated works, we ended up with 157 articles. We read the abstracts of these articles, and included for full reading the ones which satisfied the following inclusion criterion: *The work presents a software architecture to create assistive technology or solutions for accessibility.* We excluded works by using the following exclusion criteria: 1) the work does not present any software architecture; 2) the work did not present an evaluation of a software architecture or product; 3) the work could not be found or demanded payment. We ended up with 64 articles after applying the inclusion and exclusion criteria. We could not find software architectures for the provision of data of resources of mobile devices for assistive purposes, like the one we propose.

Architecture Requirements. Three two-hour sessions of brainstorming were carried with two occupational therapists and two software developers, in order to identify the requirements of the solutions that the architecture should support. Both therapists work with rehabilitation of people with motor disabilities caused by stroke. Also, the specialists reported that all of their patients own a smartphone, including those in the most difficult socioeconomic conditions. The main requirements of our architecture are: 1) the assistive solutions will consist of applications that explore the resources of mobile devices; 2) the integration of the resources must be simple to implement; 3) the applications must be easy to modify because user requirements may change over time; 4) data processing will be carried out primarily in the mobile device; 5) data storage will be be carried out primarily in a cloud-based server.

Evaluation. Since the software architecture is intended to be used primarily by software developers and software designers, we performed controlled activities in which developers could implement Android applications using our architecture.

In order to attract people to our evaluation, we offered a free 20-hour Android programming course for developers and designers who had prior experience with Java programming language and object-oriented programming. The course was held from Monday to Friday, four hours a day. From the 21 registered participants, 19 concluded the course. All the participants had smartphones equipped with Android 4.1 (or superior) for testing the applications. None of the devices had thermometer or humidity sensor, but all other hardware resources were available.

We presented our software architecture in the last day of the course. We guided the students in the creation of a mobile application for taking pictures. The action of taking the picture should be performed by both clicking a button and shaking the device. Also, the device should vibrate for the time it took for the native Android camera application to load. An additional screen of the application should show the values of the accelerometer, the gyroscope and the light sensor, simultaneously.

At the end of the course, the developers were invited to answer a questionnaire regarding the software architecture, with the following questions:

(1) Does the architecture help to create interfaces of applications that use the resources of mobile devices?
(2) Does the architecture help to create applications that uses the camera?
(3) Does the architecture help to create applications that use the microphone?
(4) Does the architecture help to create applications that use: a) the accelerometer? b) the gyroscope? c) the magnetometer? d) the proximity sensor? e) the light sensor? f) the humidity sensor? g) the thermometer? h) pressure sensor?

[1]http://dl.acm.org/
[2]http://ieeexplore.ieee.org/Xplore/home.jsp
[3]http://www.sciencedirect.com/

(5) Does the architecture help to create applications that integrate more than one hardware resource?
(6) Does the architecture help to create applications for people: a) with motor disabilities? b) with other disabilities?

All questions were answered according to a Likert scale from 1 (does not help) to 5 (helps a lot).

It is important to observe that the developers learned how to access the camera and the sensors without using the architecture in the first part of the course. They were later presented to our software architecture. Despite not using all of the resources in the test application, all the components were explained, and using one sensor is similar to using other sensors, so generalization is easy to be performed in this case.

Another phase of the evaluation was the implementation of proof of concept applications that use the resources of mobile devices for accessibility purposes. We intend to implement applications that use all of the resources that the architecture supports, for correction and improvement purposes. Two applications were created.

The first proof of concept application is intended to help users with motor disabilities to perceive and correct their body posture. In this case, the smartphone is attached to the user's chest by means of a thoracic cloth. The correct posture of the user is then calibrated. After the calibration, the user cannot exceed a body tilt threshold (in degrees) in any direction. If this happens, the application advises him/her that he/she is in a bad posture, and give audio instructions for him/her to return to the calibrated position. This application uses the three-dimensional accelerometer sensor of the smartphone.

The second application is intended to avoid pressure ulcers in people who are in bed. These ulcers are injuries to skin and resulting from prolonged pressure on the skin. Normally, they happen to people who stay for too long in the same position while in bed. By attaching the smartphone to a user's body part, we can monitor if the user has performed any significant movement in a certain period of time. If not, the application vibrates and reminds the user to change the position. This application uses the accelerometer and gyroscope of the smartphone.

4 RESULTING SOFTWARE ARCHITECTURE

Our architecture follows a client-server structure, in which the device is responsible for processing data and the server is in charge of storing data.

The sensors of current mobile devices can be classified as environmental (thermometer, pressure sensor, light sensor and humidity sensor), movement (accelerometer, gyroscope, gravity sensor and proximity sensor) and location sensors (orientation sensor and magnetometer). Our architecture provides the main functionalities of these sensors to the developer, so that he decides which functionalities and sensors to use. The full structure of the architecture is illustrated in Figure 1.

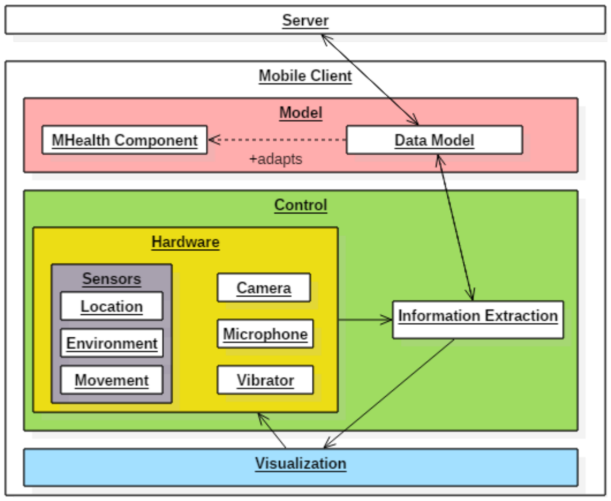


Figure 1: Software architecture Evaluated

The interaction user interface of the Mobile Client is represented by the component "Visualization." From the interface, the application accesses the control layer. In this layer, the developer can rely on XML pre-programmed files with examples of interfaces for accessing the basic data of the camera (for both taking picture and recording video), the microphone (for recording and playing audio) and the sensors (raw data). These files can be either used or ignored by the developer, for flexibility purposes.

The component "Sensors" contains Façade classes implemented in Java programming language with the main functionalities of each sensor of current Android devices. Also, it contains classes for accessing the default Android camera application, as well as supporting custom camera applications and accessing the microphone and the vibrator. The functionalities provide the developer with the possibility to retrieve the raw hardware data, or processed data. Some of these functionalities are the conversion of measurement units (e.g. (Celsius/ Fahrenheit/ Kelvin and hPa/ mBar), calculation of matrix transformations and rotation matrices, high-pass and low-pass filters, point normalization, among others. Post-processing the data resulting from the functionalities can be performed by the developer, according to the needs of the application. This process is illustrated by the component "Information Extraction". By using this module, the developer can select which sensors to be used by the application, by specifying the names of the desired sensor to a controller class, which parses the names and returns the values of the coordinates of the selected sensors. Also, all sensors wanted are registered automatically and unregistered when the application does not need them anymore.

After the information is processed by the application, the data can be stored in the "Data" layer. In order to adequate the users' information to Health parameters that might be useful for the specialists, the layer "Data" is equipped with JSON-schemas, which define the correct structure and attributes for medical parameters that are relevant in Health applications. These schemas are recommended by the open platform OpenMHealth[4], which provides a repository with recommendations of how to store medical data regarding parameters such as blood glucose, blood pressure, blood specimen type, body height, body fat percentage, body posture, body mass index, inspiratory and expiratory time, heart rate, performed activities, among others. The schemas are a recommendation and can be altered by the developer according to the needs of the application. This module provides a guide for the developer who wants to develop applications that deliver medical data to health professionals. The following example shows an archive containing an individual's body temperature, and registers that the user's maximum temperature, orally measured, between 6 a.m. and 7:23 a.m. on December 10th 2015, was 100,4 Fahrenheit.

Exemplo 1: Archive with information about an user's body temperature

```
{
    "body_temperature": {
        "value": 100.4,
        "unit": "F"
    },
    "effective_time_frame": {
        "time_interval": {
            "start_date_time": "2015-12-10T06:00:0
                0Z",
            "end_date_time": "2015-12-10T07:23:00Z
                "
        }
    },
    "measurement_location": "oral",
    "descriptive_statistic": "maximum"
}
```

5 EVALUATION

5.1 Experiment with Software Developers

The ages of the 19 participants varied from 20 to 56. Among them, 16 were under 30. Sixteen were male and 3 were female.

The answers to our questionnaire are summarized in Table 1.

The 19 participants of the course included one developer with more than 2 years of experience in Android programming and 18 developers with less than 6 months of experience in Android programming. Two participants provided answers score "2" ("the architecture helps a little"). One developer with less experience answered that the architecture "helps a little" to use the humidity or the magnetometer. Another one, also with little experience, answered that the architecture "helps a little" to create applications that use the microphone or the camera. The magnetometer, humidity sensor, pressure sensor and thermometer were the ones with the largest number of neutral responses: 4, 5, 5 and 8, respectively. This was expected since the application did no required the use of these sensors. However, this might also indicate that the architecture must simplify even more the access to these resources.

[4]Available at http://www.openmhealth.org/documentation/#/schema-docs/schema-library

Table 1: Summarizing of the responses to the questionnaire. The columns contain the number of responses in each of the Likert scales (1 to 5). The descriptions of the questions are given in Section 3.

Response related to	1	2	3	4	5
interfaces of applications	0	0	3	8	8
microphone	0	1	3	9	6
camera	0	1	3	8	7
accelerometer	0	0	1	8	9
gyroscope	0	0	1	11	7
magnetometer	0	1	4	11	3
pressure sensor	0	0	5	12	2
proximity sensor	0	0	2	11	5
light sensor	0	0	2	10	7
humidity sensor	0	1	5	11	2
thermometer	0	0	8	8	3
sensors integrated	0	0	1	9	8
motor disabilities	0	0	0	8	11
other disabilities	0	0	1	5	13
TOTAL	0	4	39	129	91
PERCENTAGE (%)	0	1.53	14.82	49.05	34.60

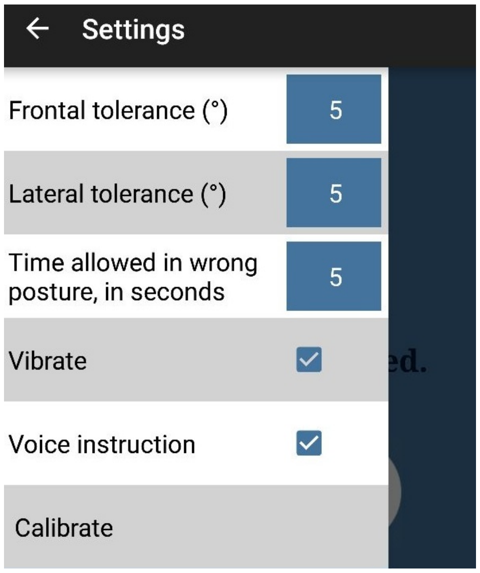
The experienced participant answered 3 in the Likert scale (neutral opinion) regarding the help of the architecture for the sensors: humidity, magnetometer, thermometer, proximity and pressure. For all other questions, his answers were 4 ("it helps").

A total 79% of the participants replied that the architecture either helps or helps a lot to create applications that use the camera or the microphone. The evaluations of the camera and the microphone were neutral (value 3) by 3 inexperienced participants and 4 (it helps) by the experienced one. The good evaluation from the experienced user might indicate that the camera and microphone components are easy to be understood by experienced developers, but since he was the only one involved in our evaluation, this assertive is inconclusive. All other answers of the participants were from 3 to 5 in the Likert scale. Two participants answered 5 (helps a lot) to all of the questions, which may represent a bias of the evaluation.

The sensor modules that received the most negative responses were the thermometer and the humidity sensor, that were not available in any of the devices used. Nevertheless, even in theses cases the percentage of responses with 4 or 5 values in the Likert scale were 57.9% for the thermometer and 68.42% for the humidity sensor. All other sensors received at least 73,68% of good acceptation by the participants. Table 2 summarizes the percentage of the responses for each question, and they indicate an overall advantage of using our software architecture.

Table 2: Percentage of responses in each Likert value.

Response related to	1	2	3	4	5
interfaces of applications	0	0	15.79	42.11	42.11
microphone	0	5.26	15.79	47.37	31.58
camera	0	5.26	15.79	42.11	36.84
accelerometer	0	0	6.56	44.44	50
gyroscope	0	0	5.26	57.89	36.84
magnetometer	0	5.26	21.05	57.89	15.79
pressure sensor	0	0	26.32	63.16	10.53
proximity sensor	0	0	11.1	61.11	27.78
light sensor	0	0	10.53	52.63	36.84
humidity sensor	0	5.26	26.32	57.89	10.53
thermometer	0	0	42.11	42.11	15.79
sensors integrated	0	0	5.56	50	44.44
motor disabilities	0	0	0	42.11	57.89
other disabilities	0	0	5.26	26.32	68.42



Frontal tolerance (°)	5
Lateral tolerance (°)	5
Time allowed in wrong posture, in seconds	5
Vibrate	✓
Voice instruction	✓
Calibrate	

Figure 2: Main screen and system parameters.

5.2 Proof of Concept Applications

Posture Monitoring System. The first step is setting the user at the best possible posture he is able to be at. This is intended to be done by a health professional, such as a therapist. Once at the right position, the application is calibrated in order to record the corresponding accelerometer coordinates.

Once calibrated, the therapist can adjust some configuration parameters, which are: frontal tilt tolerance in relation to the calibrated position (in degrees); sagittal (left/right) tilt tolerance in relation to the calibrated position; time allowed in the wrong posture; vibration feedback; audio feedback. The configuration screen is shown in Figure 2. The solution is composed by the smartphone and a chest cloth, so the application is able to monitor the torso posture posture of the user.

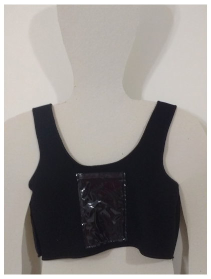
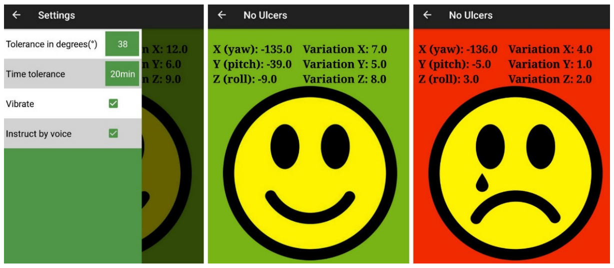


Figure 3: Chest vest used to hold the smartphone.



Tolerance in degrees(°)	38
Time tolerance	20min
Vibrate	✓
Instruct by voice	✓

n X: 12.0
n Y: 6.0
n Z: 9.0

X (yaw): -135.0 Variation X: 7.0
Y (pitch): -39.0 Variation Y: 5.0
Z (roll): -9.0 Variation Z: 8.0

X (yaw): -136.0 Variation X: 4.0
Y (pitch): -5.0 Variation Y: 1.0
Z (roll): 3.0 Variation Z: 2.0

Figure 4: Ulcer prevention application screens.

Pressure Ulcer Prevention System. Similar to the postural system, the pressure ulcer prevention system has the possibility of setting vibration and audio feedback on and off. Other configurable parameters are the time allowed at the same position and the minimum inclination tilt that identifies a change in position (the threshold).

Once the user is at the same position for the configured time, i.e., the user has not moved the monitored body part for at least the configured threshold, the application changes the color of the screen to red, displays an image of a sad face and gives vibration and audio feedback accordingly (if these options are enabled). The main screens of this applications are displayed in Figure 4

Currently, the application uses only the coordinates of the accelerometer and gyroscope. One modification under implementation is the addition of pressure monitoring, in cases which the user unconsciously lays his body over the smartphone.However, the pressure sensor is a hardware that is frequently not available in smartphones, so many of the users will not be able to take advantage of this new monitoring feature.

6 CONCLUSION

The studies reported in the literature show the importance of building applications for people with disabilities and the viability of using resources of mobile devices for building mobile applications for those people. Despite this viability, we could not find any framework or architecture that provides an easy way for developers to use the data provided by such hardware. The lack of solutions for assistive or accessibility purposes by exploring the resources of mobile devices motivated the creation of our architecture. We believe that the provision of such solution might shorten the development time of many mobile applications.

We presented a software architecture along with an evaluation with 19 software developers. The results indicate that architecture is useful for creating applications that demand using one or more resources of mobile devices. The evaluation also allowed identifying components that demand improvement. Next, we will work on improving these components and we carry out evaluations of the new version of our architecture with other Android developers.

The results of the guided tests with software developers infer that the architecture may be useful for developing not only applications for accessibility purposes that use and store medical data, but also for general-purpose applications. This seems to be specially true for inexperienced developers.

The fact that all components of the architecture may relate to each other gives the developer the possibility of using all provided components, but this is not mandatory. Thus, the developer can choose to work with whatever components of our solution, as well as adapting them to suit the application's specific needs.

Our architecture provides the use of the camera both as a main feature and as a service running on the background, and the feature is disconnected automatically when the application is no longer in use. The same applies to the sensors of the device, so less work is done by the software developer.

The widespread use of mobile devices stimulates the creation of diverse mobile applications for specific purposes by different stakeholders, both in industry and in academy. We expect that our architecture contributes to different target individuals: 1) software developers who want to create applications that rely on smartphone's resources; 2) health professionals who want to work with their patients by including computer assistive solutions; and 3) end users (both with disabilities or not) who will benefit from the created applications. Since apparently there is no such architecture reported in scientific literature, our solution may represent a new approach for developing such applications.

Our architecture provides support to the creation of applications that use camera and microphone features, thus contributing to the creation of multimedia applications. The solution will be available online, so that all people interested, from both academy and industry, will have access to it. Many of these applications may use resources of built-in components of the smartphones, such as sensors, cameras and microphones.

7 FUTURE WORKS

We propose a software architecture aiming at facilitating the development of applications for people with disabilities by using and integrating the hardware of mobile devices. The effective support of the main features of our architecture has attested by the evaluations with software developers. Our next steps include to develop new proof of concept applications by using and integrating the smartphone hardware which we did not use in the posture monitoring and pressure ulcer prevention applications.

By the results, we noticed a demand for an API for improving the way the sensors are accessed and integrated. Through this API, that is under development, the developers will specify the following parameters: sensors to be used; hardwares to be used; whether they want to take pictures or record videos; and whether camera tasks should be visual or executed in the background.

We will implement the features of taking picture and recording audio instructions in the posture monitoring application. These features will be performed by the therapist, which will take a picture of the user to record it in his/her profile and to record audio instructions that will substitute the default instructions of the application.

Finally, we will provide our architecture in an open repository such as github, along with documentation including a diagram of the situations which the architecture may be helpful.

Acknowledgments. We thank FAPESP, Capes and CNPq for the financial support of this research.

REFERENCES

[1] R. Alharthi, R. Albalawi, M. Abdo, and A. El Saddik. 2011. A Context-Aware e-Health Framework for Students with Moderate Intellectual and Learning Disabilities. In *2011 IEEE Intl. Conf. on Multimedia and Expo.* 1–6. https://doi.org/10.1109/ICME.2011.6012218

[2] R. Baguma and J. T. Lubega. 2008. A Web Design Framework for Improved Accessibility for People with Disabilities (WDFAD). In *Proc. 2008 Intl. Cross-disciplinary Conf. on Web Accessibility (W4A) (W4A '08).* 134–140. https://doi.org/10.1145/1368044.1368077

[3] Z. Chen, M. Lin, F. Chen, N.D. Lane, G. Cardone, R. Wang, T. Li, Y. Chen, T. Choudhury, and A.T. Campbell. 2013. Unobtrusive sleep monitoring using smartphones. In *2013 7th Intl. Conf. on Pervasive Computing Technologies for Healthcare and Workshops.* 145–152.

[4] A. Façanha, M. Araújo, W. Viana, and M. Pequeno. 2012. LêbrailleTWT: Providing Visual Accessibility to Twitter on Touchscreen Devices. In *Proc. 18th Brazilian Symp. on Multimedia and Web (WebMedia '12).* ACM, 313–320. https://doi.org/10.1145/2382636.2382703

[5] J. Feijó Filho, T. Valle, and W. Prata. 2012. Breath Mobile: a Software-Based Hands-Free and Voice-Free Breathing Controlled Mobile Phone Interface. In *Proc. 14th Intl. ACM Conf. on Computers and Accessibility.* 217–218.

[6] T. Hao, G. Xing, and G. Zhou. 2013. iSleep: Unobtrusive Sleep Quality Monitoring Using Smartphones. In *Proc. ACM Conf. Embedded Networked Sensor Systems (SenSys '13).* Article 4, 14 pages. https://doi.org/10.1145/2517351.2517359

[7] T. Hao, G. Xing, and G. Zhou. 2015. RunBuddy: A Smartphone System for Running Rhythm Monitoring. In *Proc. 2015 ACM Intl. Joint Conf. on Pervasive and Ubiquitous Computing (UbiComp '15).* 133–144. https://doi.org/10.1145/2750858.2804293

[8] A. Hurst and J. Tobias. 2011. Empowering Individuals with Do-it-Yourself Assistive Technology. In *Proc. of the 13th Intl. ACM Conference on Computers and Accessibility (ASSETS '11).* ACM, New York, NY, USA, 11–18. https://doi.org/10.1145/2049536.2049541

[9] N. Ohnishi, T. Matsumoto, H. Kudo, and Y. Takeuchi. 2013. A System Helping Blind People to Get Character Information in Their Surrounding Environment. In *Proc. of the 15th Intl. ACM Conf on Computers and Accessibility (ASSETS '13).* ACM, Article 34, 2 pages. https://doi.org/10.1145/2513383.2513389

[10] D. Oliveira, M. Carvalho, and T. Paixão. 2016. Ubiquitous Computing: Gestures Interaction Applied the Learning Disabilities in Process Literacy. In *Proc. of the 22nd Brazilian Symposium on Multimedia and the Web (Webmedia '16).* ACM, 107–110. https://doi.org/10.1145/2976796.2988168

[11] F. Senatore, DM Rubin, and GJ Gibbon. 2008. Development of a Generic Assistive Platform to Aid Patients with Motor Disabilities. In *Nordic-Baltic Conf. on Biomedical Engineering and Medical Physics.* Springer, 168–171.

[12] K. Upatkoon, W. Wang, and S. Jamin. 2005. WebBee: An Architecture for Web Accessibility for Mobile Devices. In *Proc. IFIP Intl. Conf. on Personal Wireless Communications.*

[13] F. Zacarias, M. Balderas, R. Cuapa, and A. Tellez. 2009. Mobile Architecture for Question Answering. In *Proc. of the 7th Intl. Conf. on Advances in Mobile Computing and Multimedia (MoMM '09).* ACM, 569–573. https://doi.org/10.1145/1821748.1821858