Sensors2PD: Mobile sensors and WiFi information as input for Pure Data

Antonio Deusany de Carvalho Junior
Computer Music Research Group
Universidade de So Paulo
dj@ime.usp.br

ABSTRACT

This article presents the results of some experiments using mobile sensors in patches of pure data. We are considering normal sensor, connected devices and Android also focusing on information as hotspots available as our best improvement in this way. During testing was performed some communication through the web server, implementing push notifications, and we apply this approach in smart phone applications to react on interactive sound installations. Ambient sound is generated using Pure Data patches, and is collectively changed in the installation environment. Sensors2PD is a generic application that can load any Pure Data patch and can be freely used for other experiments. The description of the application and installation ideas are discussed based on technical results of our tests. Our approach is analyzed as a good concept for use in mobile music performances and installations, with clear advantages for sound and music computing.

1. INTRODUCTION

Users interaction on mobile music pieces have been improved on diverse modalities. Nowadays, it is common the scenario where user receives some instructions and normally uses some interfaces, web pages, or apps conditioned to a piece. On this situation, the user notices what is going to happen if some click is being done and how the things work. Corroborating with this idea, we think that this area have great improvement during the last decade. Another point is that it is becoming hard to add different interactions from the perspective of the new devices to applications, and specifically to Pure Data Patches.

Mobile is anywhere and everywhere, so we can expect a lot of mobile devices at every concert from classical to experimental. Advances on technology has increased the number of features in devices such as sensors and transceivers, taking advantage of features of nanotechnology and processing. Another major point to keep in mind is the evolution of the Internet connection. WiFi internet access is something must have in any public place, even though in some cases the service is not so good for a crowd. Moreover, to avoid interruption of Internet access at these sites, acquiring a GPRS/3G/4G plan is a condition for being always on. The devices are always looking for the best way to communicate over the Internet, and nowadays WiFi connections are chosen with a higher priority, because of its speed and range, considering that users save a lot of WiFi networks with automatic connection option checked. It is important to note that mobile devices continues searching (best) available Internet connection , even when connected.

Thinking of presentations of mobile music, lots of sensors and specific local information on mobile devices can be used in applications. Using sensors is attractive considering its variety and sensitivity range. Besides sensor importance, we became interested in the use of WiFi information that is reported over time. We also used the continuous Internet connection to exchange text information between users through web server.

Some works have already evaluated the use of WiFi Indoor location for robots [1, 2] and had good results: 1 meter accuracy, approximately. This observation can be useful for presentations inside where GPS will not work, and the use of WiFi information can not be easily perceived by users, if the mobile device is always in search of mobile networks, as already pointed. The fact that the phone is always connected the network is another important information that had taken our attention especially because during some performances we shared some information among participants through Wireless and had improvements in interactions without user perception too. Surely, heavy consumption of 3G for large flow of information can be a problem for some people, then it should be better to use short messages and send them sporadically to avoid running out Internet User data occurs. This approach may be useful for sending information and receiving notification through a web server and create new forms of interaction between users during presentations of mobile music.

We are going present the way we developed Sensors2PD, how easy we can send sensors informations to Pure Data patches, and explain the other possibilities we can get if we use sensors of WiFi and web servers on performance-related applications. In the end there will be an explanation about experiences with the application.

2. RELATED WORK

This area has a bunch of works related to mobile music installations for user interaction using sensors and web servers with mobile devices. The main focus of some works are becoming similar, and they discuss that HTML5 and Javascript on web browsers are the best solution for this kind of inter-

action. In SWARMED [3], the authors developed a framework for audience interaction during an installation accessing a website on captive portal. The novelty of using a captive portal is interesting on the aim of focusing user's Internet access only to the performance application during the presentation. On the other hand, the author notice that each participant would like to hear its contribution to the performance, but all the sounds are reproduced only on the central performer, and it is hard to give this feedback if lots of people is participating.

On NEXUS [4] project, that is related to distributed performance, authors attempt that a cross-platform application can really be conceived using dynamic web page, allowing server side web application to do the hard work while the participants interact using an interface placed into a browser on its own device. Even if both works use Ruby on Rails, NEXUS leaves the application on line for distribution and scalability facilities aiming a large number of participants from different places.

On the other hand, we have urMus [5], an environment for mobile application development. This is a directly related work that has an event-handling for mobile sensor events, with lots of features that permits easy musical application development. It is possible to load scripts and also share code between users over the network in some applications developed with urMus[6]. The only limitation for musicians is that Lua isn't used as Csound and Pure Data.

Another related work uses Csound as main *patch* language [7]. The users loads Csound orchestras and scores on mobile devices and uses interface options and some sensor events for interaction. This approach was a motivation for this work, and we also notice some advantages of a similar project, PdDroidParty [1] as inspiring for a new way of musical interaction using a common language for musicians and some transparent information presented below.

3. TRANSPARENT INFORMATION

There are lots of transparent information that can be used on mobile applications, and we are going just to cite some of them briefly:

- Available WiFi networks can be found everywhere, and the users are living the Always-On Era;
- The cost for send and receive short informations from web servers is becoming unimportant bearing in mind the price of 3G plans on most part of the world;
- Push notifications are being used by the most of the applications installed on mobile devices, aiming to keep the user updated, but always consuming users Internet connection as much as possible;
- The use of hot-spot informations to facilitate localization has been a used first with robots, but also by general systems like Google Maps.

Thinking about those premises we can conclude that we can extrapolate the usual limits of performances on a theater, with lots of mobile users, for example, considering "transparency" in the user point of view. Of course the user need to know that we are using those information, but it does not need to be explicitly during the performance.

4. SENSORS2PD

The integration of PD with Android application became possible with libpd, a library that grant loading patches and exchange messages using senders and receivers on applications. One of the drawbacks of this integration is a slight need of Java programming experience intended for Android application development. Apart from that situation, some native applications have been developed to smooth the way for non (Java) programmers use its own patch on Android devices. The most famous application with this concept is PdDroidParty. This application searches internal storage for patches and permits interaction using some interface options. Although it is a great solution, it does not facilitate the use of sensors with patches.

Android devices possess lots of sensible sensors that have been used on many situations. We notice that the quantity of sensors and its range vary based on device and also Android API, resulting in some restrictions during mobile performances if you want to use specific sensors or have different devices. Our approach try to solve these problems and presents other sensor's information that can be used.

4.1 Using sensors on Pure Data patches

Sensors2PD is an application that makes possible the use of all Android sensors and even more useful data from device functionalities. The application uses libpd [2] to load the patch that users can seek and select from mobile storage. Receivers for any sensor value can be used on the patch following the specification found on the guide. Depending on the sensor id (ID) and variable number (#), you need to configure the receiver like that:

[r sensorIDv#]

If you want to use Accelerometer, which has ID=1 and 3 variables, you can receive the sensor values using:

[r sensor1v0]

[r sensor1v1]

[r sensor1v2]

This is a reasonable approach considering that the application will be listening for any sensor variation and might send its updated value to the patch as fast as possible. Android sensors can be listened at four different rates, varying from $\tilde{0}$ to 200ms. The fast you retrieve values the fast you drain the battery on mobile application, but during some tests we've found out that it is not a problem to use the fastest mode if the normal time of a performance or installation won't be more than one hour. Despite the fact that not all the sensors are supported on all devices, it is not a

[1] PdDroidParty: `http://droidparty.net`

[2] libpd: `http://libpd.cc`

great problem. Only available sensors are going to be listened, and only the receiver for an unavailable sensor will not receive any value.

We considered each functionality that detects physical variations as a sensor, so other kind of sensors are being used on this application and can also be used with PD patches.

4.1.1 Touch events and position

All Android devices support multitouch interaction nowadays with high sensibility as it is the principal and the most intuitive way to access mobile options. We can manipulate up to 14 simultaneous touch ids to get noticed about its events and position on x and y axes, even if most mobile devices detect only 5 or 10 fingers. Based on the same procedure defined before, it is possible to use the position from every touch your device screen can detect. The receiver on the patch will need touch id (ID) and position coordinate x or y (#).

[r sensorTIDv#]

If you want to track first touch position, which has ID=0, you need:

[r sensorT0vx]

[r sensorT0vy]

4.1.2 Audio input

Audio input using [adc~] on Pure Data can have better results only when Pure Data is used as an Service on the application, and then we have opted to run libpd as background Service on our application. It is necessary to bear in mind that background operation continues to run even when you left the application or change to another one, like answering a call. In spite of this fact we preferred not to do anything on this case to prevent some installations to stop working while the application is running.

We do not need to worry about audio input and output between device, application, and PD while using libpd. On the other hand we can't control the audio configurations due to device specifications, so there is no guarantee about sample rate, block size, or number of channels. Notwithstanding that this can be important for some patches, we notice that audio quality was awesome on our tests, except for the delay between input output even on loop-back.

4.1.3 WiFi networks

Conceived by the concept of transparent information, we've decided to use the sensor of hot spots as another option for the application interaction. The sensor of hot spots gives lot of information as SSID, frequency, and level in db from network found. On our application we've decided to send the level as information for the PD patches. You need to configure the receiver with the SSID name (ID), that can't have any spaces.

[r sensorW-ID]

If your WiFi SSID is "MyRouter", you'll need the receiver:

[r sensorW-MyRouter]

Regarding that the WiFi level varies normally between -100 dBm to -1dBm depending on how far you are from the hotspot, you can use this value on many applications for indoor localization based on works from other areas of literature like robotic [1, 2].

The WiFi network signal has lots of problems with interference that turns the signal always unstable, but it still can be used as positioning system. There are 11 different frequencies for hot spots and the antenna needs to search each channel before returning a list of networks. Each request of available networks took 500ms to 2s on our tests and we noticed that some hot spots have minor problems during these search because of the orthogonality of channels 1, 6 and 11 frequencies.

4.1.4 Other sensors input under development

There is also other sensors being tested and implemented to add more possibilities:

- Bluetooth using SSID
- GPS using Google API
- Color identification using camera and OpenCV

We also consider that this application can be easily ported to other mobile technologies like iOS and desktop application. The integration of this application with other systems is not hard, and we also use some communication through web servers to send some sensors values to other applications. Informations about the web server are going to be described below.

4.2 Web server integration

Considering the intention to send and receive information between devices, the easiest way to implement was using web server. The development of web server have been intensified over time so that new methodologies as Rapid Application Development (RAD) began to be practiced with the use of tools and specific frameworks. One of the technologies that emerged following the RAD methodology was Ruby on Rails (RoR) [3] . Rails is an open source framework for web applications that supports Ruby language.

We have chosen JSON format as the preference for communication with RoR web server. JSON is an open standard for exchanging data that is based on a text format understandable by humans without technical knowledge of computer science. The format is derived from the Javascript language and presents itself as an alternative to XML format. Sharing content with web server using this format comes to be advantageous also with the ease of encoding and decoding messages.

Those technologies are generically disposed to any kind of content, and its use in the music can open several possibilities for sharing content and predispose infinite ways of musical interaction [8, p. 123].

[3] Ruby on Rails: `http://rubyonrails.org/`

5. EXPERIMENT AND INSTALLATION

We proposed an experiment to test the Sensors2PD at an interactive sound piece. We have created Pure Data patches using lots of sensors as input, and the users were advised to walk around a place to find some hot spots and interact with them. The sound output amplitude on mobile devices was controlled by the WiFi signal of the hot spots.

Other sensors were used to change sound velocity, pitch, articulation, and other expressive values. During the experiment, the application have sent hot spot information to the web server, so the hot spots knew how many users were near and could play different songs to interact with the public, using push notifications and transparent informations. Figure 1 shows people during the experiment.

Figure 1. People appreciating the installation

5.0.1 Technical discussion about Internet access

When a device is configured as hot spot, it can't check for available networks, but it is possible to access the Internet using other technologies: 3G, Bluetooth, Ethernet. It is important to notice that mobile devices differ in signal level when set as a hot spot due to their different antennas. Before this installation, we needed to check out the distance between hot spots to avoid the user find more than one good signal at the same time. In this case, we programmed to start the interaction only when the user find a hot spot with level greater than -40dBm.

6. CONCLUSIONS

The development of Sensors2PD predispose other users to create applications using Android sensors with Pure Data patches. As an original solution, we can say that the community can help to improve considering its distribution as open source code, and maybe it can be integrated to other applications as PdDroidParty and so many others. The ideas of installation presented here and using this application can be characterized as a start point to other installations and we wish the community can take advantages of our contribution as much as possible.

The use of sensors, transparent information, and web server push notification approach was evaluated with good results. There were lots of questions from users during the installation, and most of doubts were related to some interaction aspects that were incomprehensible for them at first sigh, due to the transparent information used. This condition shows some drawbacks and advantages of our concepts, indeed.

7. ACKNOWLEDGMENTS

We would like to thank CAPES and FAPESP for funding. The support of NuSom Research Centre on Sonology [4] and Computer Music Research Group [5] is really appreciated. We thank Marcelo Queiroz, André Bandeira and Flávio Schiavoni for suggestions and ideas. We also thank Antônio Barreto for his time and willingness to help.

8. REFERENCES

[1] J. Biswas and M. Veloso, "Wifi localization and navigation for autonomous indoor mobile robots," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 4379–4384.

[2] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of wifi based localization for smartphones," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 305–316. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348581

[3] A. Hindle, "Swarmed: Captive portals, mobile devices, and audience participation in multi-user music performance," in *Proceedings of the The International Conference on New Interfaces for Musical Expression*, ser. NIME '13, 2013.

[4] J. Allison, Y. Oh, and B. Taylor, "Nexus: Collaborative performance for the masses, handling instrument interface distribution through the web," in *Proceedings of the The International Conference on New Interfaces for Musical Expression*, ser. NIME '13, 2013.

[5] J. W. Kim and G. Essl, "Concepts and practical considerations of platform-independent design of mobile music environments," in *Proceedings of the International Computer Music Conference*, 2011, pp. 726–729.

[6] S. W. Lee and G. Essl, "Communication, control, and state sharing in networkedcollaborative live coding," in *Proceedings of 14th International Conference on New Interfaces for Musical Expression (NIME)*, Goldsmiths, University of London, London, UK, June 2014.

[7] V. Lazzarini, S. Yi, J. Timoney, D. Keller, and M. Pimenta, "The mobile csound platform," in *Proceedings of the Internationa Computer Music Conference*, 2012.

[8] F. Iazzetta, *Mica e Mediaoo Tecnolgica*. Perspectiva, 2009.

[4] NuSom: `http://www2.eca.usp.br/nusom/`

[5] Computer Music Research Group: `http://compmus.ime.usp.br/`