Toward a Web of Audio Things

Benjamin Matuszewski, Frédéric Bevilacqua

Toward a Web of Audio Things

Benjamin Matuszewski
CICM/musidance EA1572, Université Paris 8,
UMR STMS IRCAM-CNRS-UPMC
Paris, France
benjamin.matuszewski@ircam.fr

Frédéric Bevilacqua
UMR STMS IRCAM-CNRS-UPMC
Paris, France
frederic.bevilacqua@ircam.fr

ABSTRACT

Recent developments of web standards, such as WebAudio, WebSockets or WebGL, has permitted new potentialities and developments in the field of interactive music systems. Until now, research and development efforts have principally focused on the exploration and validation of the concepts and on building prototypes. Nevertheless, it remains important to provide stable and powerful development environments for artists or researchers. The present paper aims at proposing foundations to the development of an experimental system, by analysing salient properties of existing computer music systems, and showing how these properties could be transposed to web-based distributed systems. Particularly, we argue that changing our perspective from a *Mobile Web* to a *Web of Thing* approach could allow us to tackle recurrent problems of web-based setups. We finally describe a first implementation of the proposed platform and two prototype applications.

1. INTRODUCTION

In last years, the specification and implementation of new Application Programming Interfaces (APIs)—such as WebAudio, WebSockets or WebGL—in web browsers has allowed for envisioning the web platform as a fertile playground for artists and musicians. [1] The inherent networked nature and scalability of web technologies has permitted to simplify and democratize the use of mobile devices (e.g. smartphones) in the diffusion of sounds. [2] As such, these web-based distributed systems can be considered as a new development in the long history of multi-source electroacoustic music that starts with the creation of the *acousmonium*, created by Francois Bayle and Jean-Claude Lallemand at the *Groupe de Recherche Musicale* in 1974. [3]

Moreover, recent works have demonstrated that sufficient synchronization can be achieved for distributed audio rendering in space. [4] Altogether, these technologies open new opportunities for the exploration of intertwined sonic and social interactions between participants [5] (see Fig. 1) or between a performer and a participative audience. [6]

Figure 1. *Collective Loops* - An installation based on web technologies featuring synchronized audio-visual rendering over smartphones and light projection on the floor.

These works, among others, have demonstrated the qualities and potential of these technologies in terms of interoperability. However, we can observe some recurring difficulties, particularly in supporting exploratory processes. This paper proposes to tackle this issue in improving existing frameworks and setups. Specifically, based on our past experience, we highlight the following issues:

- Experimenting and exploring in fast iteration cycles —crucial to artistic and research contexts—remains difficult due to constraints inherent to the usage of smartphones.
- The lack of versatile and high-level tools targeted towards non-expert users (e.g. artists or researchers). Currently, experimenting with web technologies implies to first learn the basic involved technologies (e.g. HTML, JavaScript, server-side implementation).

The methodology proposed in this paper is to first analyse conceptual and design aspects that characterize successful experimental computer music platforms. Then, we propose to extend our formalization of such systems to *rethink* the development of web-based musical experimental system. Conceptually, our proposal can be thus considered as a generalization built on top of existing libraries and frameworks. [4, 7, 8] As we will fully explain in the second part of the paper, this generalization corresponds to move from a *Mobile Web* approach to a *Web of Things* approach. [9]

Precisely, our main contributions in this paper are threefold:

- Formalizing a conceptual model for conceiving and developing experimental music systems, enabling col-

laborative work between artists and researchers (section 2).

- Proposing a change of perspective, from a *Mobile Web* approach to a *Web of Things* approach, enabling novel features facilitating the appropriation of current setups (section 3).

- Developing building blocks along with two implemented prototypes applications (section 4).

We believe that our proposal will help to deliver a more effective experimental platform dedicated to the prototyping of distributed, embedded and interactive music. As such, it could offer an interesting playground for multiple artistic and research areas (e.g. collective and collaborative human-computer interactions, spatialization, multi-agents systems or distributed computing), as well as improve the collaboration between different actors such as composers, developers and researchers.

2. CONCEPTS

2.1 Experimental Platform in Art and Science

Recent studies on artistic research, [10] based on the Rheinberger's epistemological work on experimental systems, offer renewed perspectives on known issues in methodological and epistemological point of views. [11, 12]

In Rheinberger's conceptual model, the experimental system is the starting point of every research work as well as a dynamic system composed of *technical things*—that embody what is already known—and of *epistemic things*—which are still to be known and stabilized in a technical artifact. As such, an experimental system must exhibit and—maybe more importantly—preserve two fundamental and conflicting properties: it must be well enough defined to be manipulated properly, while being sufficiently open and unstable to enable serendipity and emergence of new epistemic things.

As illustrated in Fig. 2, we can thus consider artistic and scientific research as processes that share similar structural properties—despite qualitative differences in the outcomes—in which the experimental platform can act as a shared space. Indeed, in both cases, the "completion of the research process is marked by the stabilisation of the epistemic object as a fact" [10]—be it a publication or an artwork—which has to be evaluated and validated before being re-injected into the experimental system as a stabilized knowledge.

Until now, research work dedicated to distributed and web-based music systems has focused on exploring and probing their inherent possibilities and opportunities, rather than in the characterization and implementation of an experimental platform that could act as a shared space for artists and researchers. We propose to examine existing computer music systems to identify salient design properties that a web-based experimental platform should embody.

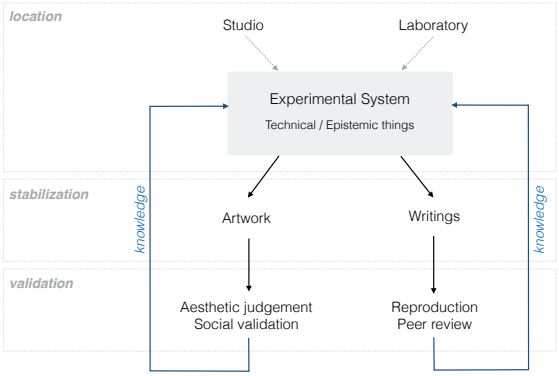

Figure 2. Figure that highlights how the experimental platform can be considered as a shared space that embodies knowledge from different type of research outcomes.

2.2 Design Properties of Computer Music Systems

Music programming systems such as *Max/MSP* and *PureData* [13] [14] are nowadays among the most popular and efficient prototyping and experimental platforms dedicated to electroacoustic music. Despite the fact that these environments are not specifically designed to tackle problems inherent to distributed music systems—such as synchronization or distributed state—nor to be integrated in a web based framework, we can consider that some of their properties are particularly interesting for approaching the development of a web-based experimental platform.

More precisely, we postulate that their success relies in part to their effective implementation of specific patterns that greatly facilitate the user to accomplish particular tasks. To explain such a statement, let's first recall two important concepts from software design: *extensibility* and *composability*.

2.2.1 Extensibility and Composability

In software architecture, *extensibility* is a common systemic design pattern that describes the possibility of adding or modifying functionalities of a program without impacting its dataflow or internal structure. From a user-centered perspective, it allows the user (e.g. composer, musician) to include new functionalities—implemented and shared by others—seamlessly into its own workflow.

The *composability* pattern—closely related to *extensibility*—defines the ability of a system to provide components that can be selected and assembled in multiple ways. In *Max/MSP* or *PureData*, this property is implemented by providing to the user a Domain Specific Language (DSL) in the form of a Visual Programming Language. More precisely, these environments expose a number of *boxes*—each one implementing some very specific and possibly high-level functionality—that can be linked together in multiple ways [15]. Hence any user (e.g. artist, researcher, creative coder) can define the final behavior of the program by composing a graph of boxes.

As shown in Fig. 3, these music programming environments implement *extensibility* and *composability* by establishing a clear separation among users and/or tasks, such as

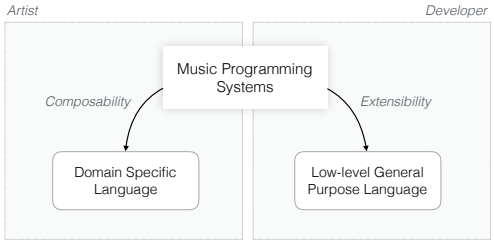

Figure 3. Figure that highlights how traditional Computer Programming Systems, such as *Max/MSP* and *PureData*, implement the *composability* and *extensibility* patterns by establishing a clear distinction between two types of users.

low-level programming or high-level patching. On the one hand, an important part of the *extensibility* of the system is realized by using a low-level General Purpose Language (e.g C/C++); in turns, this part of the system becomes accessible only to users that have expert and very specialized programming skills. On the other hand, *composability* is achieved by exposing a number of (*black-*)boxes to the high-level user. [1]

2.2.2 Immediate Feedback and State of Flow

We argue that the most important characteristic, that makes these environment such efficient experimental and prototyping platforms, appears when formalizing the *immediate feedback* property. Indeed, unlike many programming tasks where the problem to solve is well defined, artistic and research practices can be considered as exploratory tasks were the problem to solve—the *epistemic thing*—is intrinsically ill-defined.

Therefore—and more critically when programming in an experimental setup composed of multiple devices distributed in space—a conflict appears between the time devoted to write and update the application and, the need to maintain the required state of flow [16] for working on an exploratory task. In current web-based setups, this conflict is generally solved by providing the end user (e.g. artist, researcher) a closed application that limits *de facto* its agency and ability to explore possibilities not implemented by the developer in the first place.

Max and *PureData* successfully resolved these issues by providing a versatile programming environment where any modification leads to an auditory or visual feedback in real-time, and that allows non-expert programmers to mostly forget about many complex tasks (e.g. creation of Graphical User Interfaces) that are not related to their particular domain or current activity.

3. FROM MOBILE WEB TO WEB OF THINGS

We will now present how the concepts described in the previous section apply to Web technologies. First, Web technologies, in the context of artistic practices, hold the promise to blur the differences in the environments used by developers, artists and researchers, since they are based on a common platform and a common scripting language (i.e. JavaScript). This common environment is specified by a consortium, the *W3C* (*World Wide Web Consortium*), and implemented in web browsers by majors companies such as *Google* or *Mozilla* (which tends to guarantee a certain stability). In this context some of the lower-level aspects of the *extensibility* of the system are thus managed by external actors (see Fig. 4). In turn, this situation provides a full featured scripting language that allows users with different background and objectives to work at various levels of the same system. For example, a composer or researcher could intervene directly in the implementation of a domain specific functionality without facing the necessity of learning a new language nor need to have a specialized developer on its side.

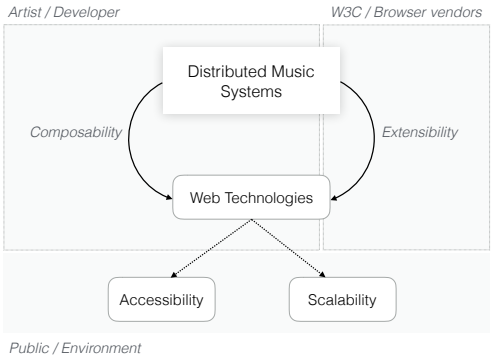

Figure 4. Figure that shows the changes, in regard to Fig. 3, introduced by the distributed aspects and Web technologies in Music Programming Systems.

3.1 From Mobile Web...

3.1.1 Definition and Potentialities

Web technologies used conjointly with mobile devices (e.g. smartphones) has been shown powerful to design musical interactive systems. [5] We can refer to this approach as a *Mobile Web* approach, or in other words, the possibility of accessing browser-based applications from handheld devices through wireless networks (e.g. WiFi).

Indeed, such systems benefit from the ubiquity of mobiles devices. Moreover, the use of web technologies importantly ease the cross-platform deployments, avoiding current limitations in publishing native applications.

3.1.2 Drawbacks

However, such setup hardly manages to deliver an effective and efficient experimental platform, especially regarding the need of *immediate feedback* described earlier. Indeed, using smartphones as main components of the technical setup impedes prototyping and experimental processes by several aspects.

Principally, the complexity and difficulty of the workflow imposed by closed source operating systems and manufacturers, that do not allow to automate some of the more

[1] Such DSL—even in the Visual Programming Language paradigm—necessitate some learning and skills that generally cannot be reused in other domains.

repetitive tasks. For example, the need to manually perform numerous tasks such as: configure and launch the application on each device independently, necessity to touch the screen to start the audio playback each time the application is reloaded, etc. Another important aspect is the cost of such devices that hinders the possibility to create such prototyping setup in the first place.

3.2 ...To Web of Things

To tackle these issues, we propose to move from a *Mobile Web* to a *Web of Things* approach. We argue that this new perspective could resolve some of the problems described above, and thus allow to define and build a more efficient experimental platform.

3.2.1 Definition

The notion of the *Internet of Things*—closely related to the computer science field of *pervasive computing* or *ubiquitous computing*— has been defined in the early 90s at the Xerox PARC Research Lab by Mark Weiser in its seminal paper *The Computer for the 21st Century*: [17]

> The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. [...] Therefore we are trying to conceive a new way of thinking about computers in the world, one that takes into account the natural human environment and allows the computers themselves to vanish into the background.

In recent years, the reduction of size and cost of microcontrollers has allowed the development of such *things* in many application domains such as smart home and cities, industry or wearables. In this context, the *Web of Thing* approach, [9] together with the spread of microcomputers, proposes to simplify the development, deployment and interoperability of multiple devices by relying on Web standards.

3.2.2 Improving Immediate Feedback

In the domain of distributed music systems, the use of tiny computers—such as the Raspberry PI—could strengthen the current technical setups in several ways, particularly regarding the *immediate feedback* property described earlier (see Fig. 5).

Additionally to the reduced cost of these devices, some of their properties make them particularly interesting in our objective:

- They run under a Linux operating system (which makes them quite simple to install and setup), have a low energy footprint and out of the box WiFi capabilities.
- They have enough processing capabilities to run a *Node.js* environment, allowing to use the same programming language (i.e. JavaScript) on every part of the system. Importantly, as described in the next section, this allows us to reuse existing powerful libraries for building collective interactions using web technologies.
- They can be easily scripted to automate repetitive tasks, such as relaunching an application whenever a change occur in the codebase.

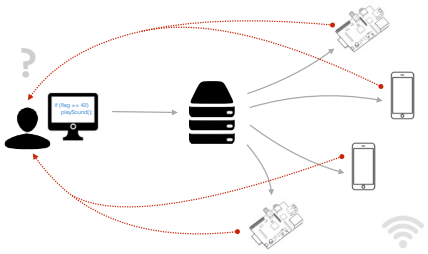

Figure 5. Figure that illustrates the envisioned distributed prototyping platform where any modification leads to an immediate feedback from the test devices (smartphones and/or embedded devices) to the user.

3.2.3 Opportunities

Using such microcomputers also comes with the possibility of extending each module with dedicated sensors and actuators. Together with the scalability offered by Web technologies, such technical setup—composed of numerous modules—could enable new possibilities of expressive interactions between people, environment, tangible interfaces and spatialized sounds.

Furthermore, the more controllable and standardized aspect of the hardware and software could help to create a bridge—both technological and pedagogical—with more traditional computer music approaches and tools.

4. BUILDING BLOCKS & PROTOTYPES

> The technology required for ubiquitous computing comes in three parts: cheap, low-power computers that include equally convenient displays, a network that ties them together, and a software systems implementing ubiquitous applications. [17]

In the design of the system, we choose to build from modular and exchangeable elements over an all-in-one but more monolithic approach. This architecture results of tradeoffs between: compatibility with current web based systems, scalability, simplicity of maintenance, sustainability and openness to evolutions. These points appear to be of primary importance, particularly in a technical context where hardware and software evolutions occur at very high rate.

4.1 Hardware

A single *module* is composed of the following minimal hardware setup:

- A microcomputer system, such as the Raspberry PI Model 3, a popular microcomputer featuring in-built wireless abilities such as WiFi and bluetooth, a Quad Core 1.2GHz 64bit CPU and 1GB of RAM, running the Raspbian Stretch Lite operating system.
- A soundcard with 2 channels inputs and outputs.
- A battery for embedding a module into the environment with a relative autonomy, at minima for the duration of a concert.

The choice of such common hardware relies in large part in the necessity to minimize price and maintenance of a fleet composed of numerous devices. Additionally, it allows to replace or update each component independently which improves sustainability, and leaves the door open for extensions with sensors and actuators.

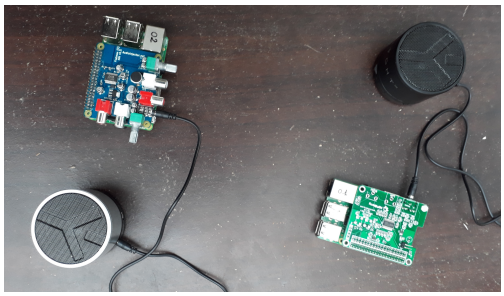

Figure 6. Figure that shows two modules composed of a Raspberry PI 3, a soundcard and a small speaker.

4.2 Software Prototypes

On the software side, we explored two concurrent strategies for the implementation of our *modules* as clients of a distributed system. Both approaches are based on a JavaScript environment, enabling the use of the same programming language on every element of the system (eg. server, mobile or thing clients). As such, they also allow to reuse—or simply adapt—many existing software components such as clock synchronization, [4] scheduling [8], stream processing [18] or, at the application level, functionalities offered by existing frameworks. [7]

To explore the strengths and drawbacks of each solution, we implemented two simple generative music systems based on an array of the hardware modules described above. [2]

4.2.1 Etude 1: Pteroptyx Malaccae

Pteroptyx Malaccae is based on the firefly synchronization algorithm described in [19]. In this prototype, each module embodies a single firefly that tries to synchronize with its peers. The emergent synchronization process is sonified using short burst of noise. All modules also contribute to a common sonic environment by playing a part in a distributed additive synthesis.

Technically, each module runs the Web application using a headless *Chromium* browser, and thus, acts as generic client of the system. In term of operability, this approach proves to be very convenient as *things* follow thus the exact same technical paradigm as *mobile* clients.

However, this solution also comes with all the constraints and limitations common to the sandboxed nature of web browsers. For example, it prevents from accessing the file system or using sensors and actuators in a simple and convenient way. Indeed, doing such tasks would imply to run a parallel application on each module and to create a bridge for communicating between the *Chromium* browser and this dedicated application.

4.2.2 Etude 2: Clock(s)

Clocks illustrate the synchronization possibilities enabled by simple adaptation of existing libraries. In this prototype, each module plays a simple periodic rhythmical pattern built on a single pitch. The sonic result can be considered as a distributed polymetric and polyrhythmic texture where complexity arises from the number of devices and their distribution over space.

From a technical point of view, the prototype is based on a *Node.js*—a JavaScript environment built on top of the *Google Chrome*'s V8 engine—application. The environment however, does not yet provide an implementation of the WebAudio API for audio rendering. To overcome this problem, we developed a native *Node.js* extension [3] [4] on top of the *lib-pd* library. [20] This new functionality enables audio rendering by allowing to open and control *PureData* patches directly from the JavaScript code.

However, this solution also comes with its own strengths and drawbacks. On the one hand, using *Node.JS* applications as clients of the system allows for the simple usage of all the capabilities of the underlying platform. For example, a number of third-party libraries are dedicated to the usage of sensors and actuators on microcontrollers. [5]

On the other hand, this approach implies, for now, the introduction of a technology that is not part of the Web APIs and, as such, limits the possibilities of code reuse and cross-platform developments concerning audio synthesis. Nevertheless, the usage of the *lib-pd* library in this context also propose an interesting and proven pattern for the decoupling of application logic, scheduling and audio rendering that could serve as model for future developments.

4.2.3 Performance Considerations

As described in the two previous examples, each of the envisioned approach—*Chromium* or *Node.JS*—comes with its advantages and limitations. To acquire a better picture of their respective implications in term of audio latency and memory footprint, we created a simple synchronized metronome application implemented following both approaches. [6]

[2] The source code of each prototype is available at `https://github.com/b-ma/pteroptyx-malaccae` and, `https://github.com/b-ma/clock-s-`.

[3] In the *Node.js* environment, the JavaScript runtime can be extended with new functionalities using native languages such as C/C++ or Rust.

[4] `https://github.com/b-ma/node-libpd`.

[5] `http://johnny-five.io/`, `https://cylonjs.com/`.

[6] The measures presented are only meant to give an idea of the order of magnitude of the performances of each approach. In any case, they can be considered as accurate or precise results.

Concerning latency, we could observe a delay of around 70*ms* introduced by the *Chromium* version compared to the *Node.js* / *lib-pd* version of the application.

In term of memory footprint, a slight difference seems to appear in favor of the *Node.js* approach too. Indeed, while the CPU usage is around 10% in both cases, *Chromium* uses around 20% of RAM, while the *Node.js* process only uses around 5% of RAM.

Overall, despite the current unavailability of the WebAudio API in a *Node.js* context, this approach seems to exhibit more qualities and interesting properties than the browser-based one. Additionally, we can reasonably expect that an implementation of the WebAudio API will be available in this environment in a near future. As such, it should be considered as an interesting basis for our future developments.

5. DISCUSSION AND CONCLUSION

In this paper, we have proposed a system dedicated to web-based distributed music designed from the point of view of the artist or researcher in situation of exploratory tasks. Starting from the notion of experimental system, we described some of the design properties that characterize existing computer music systems and their articulation within our specific context. We then proposed to generalize the current *Mobile Web* approach to a *Web of Things* approach as a mean to overcome some of the drawbacks of existing prototyping setups. We finally described a first implementation of the proposed solution, and two prototype applications to assess its feasibility.

Concerning the use of microcomputer, with Linux operating system, our proposal can be compared to two existing platforms, the *Bela* [21] and the *Satellite CCRMA*. [22] Both projects are designed toward the implementation of new musical instruments, embedded projects and installations.

The *Bela* is an environment focused on low latency based on the *BeagleBone Black* microcomputer. The environment is a combination of an extension board dedicated to audio and sensors processing and a Linux distribution featuring a real-time kernel.

The *Satellite CCRMA* is a project based on the composition of a Raspberry PI microcomputer and an Arduino Nano microcontroller. The kit also comes with a specific Linux distribution (i.e. the *Satellite CCRMA distribution*) specifically configured for interactive media.

Both projects share the same philosophy of proposing a dedicated Linux distribution. However, the example of *Satellite CCRMA* shows that this strategy introduces a fragility in terms of sustainability and maintenance. Indeed, the last published version of the *Satellite CCRMA distribution* targets the Raspberry PI 2 and has not been updated since then. In contrast, our strategy tries to minimize this risk by accepting drawbacks of standard—and exchangeable—hardware and software (e.g. operating system), and by building on top of widespread technologies.

We believe that our proposal can serve as a basis for the implementation of an effective experimental platform dedicated to web-based distributed and interactive music systems. As such, it could simplify and democratize the usage of these technologies—while retroactively improving them—in artistic and research contexts. However, to successfully achieve this goal, many points still need to be tackled.

First of all, a first set of possible hardware setups need to be stabilized and characterized. An application dedicated to the deployment and management of the fleet of modules must be implemented. Parallely, a set of high-level tools dedicated to the simplification of the experimentation (e.g. GUI tools) will have to be defined and implemented. Each of these tools will need to be designed iteratively using feedbacks from non-expert developer users such as composers, researchers or computer music designers.

Acknowledgments

We would like to thank our colleagues Norbert Schnell, Jean-Philippe Lambert, Diemo Schwarz and Joseph Larralde for their precious contributions to this work.

6. REFERENCES

[1] L. Wyse and S. Subramanian, "The Viability of the Web Browser as a Computer Music Platform," *Computer Music Journal*, vol. 37, no. 4, pp. 10–23, 2013.

[2] L. Wyse, "Spatially Distributed Sound Computing and Rendering Using the Web Audio Platform," in *1st Web Audio Conference. Paris*, 2015.

[3] B. Tayler, "A History of the Audience as a Speaker Array," in *Proceedings of the NIME17 Conference*, 2017.

[4] J.-P. Lambert, S. Robaszkiewicz, and N. Schnell, "Synchronisation for Distributed Audio Rendering over Heterogeneous Devices, in HTML5," in *Proceedings of the 2nd Web Audio Conference*, Atlanta, US, 2016.

[5] N. Schnell, B. Matuszewski, J.-P. Lambert, S. Robaszkiewicz, O. Mubarak, D. Cunin, S. Bianchini, X. Boissarie, and G. Cieslik, "Collective Loops Multimodal Interactions Through Co-Located Mobile Devices and Synchronized Audiovisual Rendering Based on Web Standards," in *Proceedings of the Eleventh International Conference on Tangible, Embedded, and Embodied Interaction*. Yokohama, Japan: ACM, 2017.

[6] B. Matuszewski and N. Schnell, "GrainField," in *Proceedings of the Audio Mostly Conference*. London, UK: ACM, 2017.

[7] S. Robaszkiewicz and N. Schnell, "Soundworks a playground for artists and developers to create collaborative mobile web performances," in *Proceedings of the 1rst Web Audio Conference*, Paris, France, 2015.

[8] N. Schnell, V. Saiz, K. Barkati, and S. Goldszmidt, "Of Time Engines and Masters An API for Scheduling and Synchronizing the Generation and Playback of Event Sequences and Media Streams for the Web Audio API," in *1st Web Audio Conference. Paris*, Paris, France, 2015.

[9] D. Guinard and V. Trifa, *Building the Web of Things*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2016.

[10] M. Schwab, *Experimental Systems: Future Knowledge in Artistic Research*, ser. Orpheus Institute series. Leuven University Press, 2013.

[11] C. Frayling, *Research in Art and Design*, ser. Royal College of Art research papers. Royal College of Art (Great Britain), 1993.

[12] M. Schwab, "Between a rock and a hard place," in *Intellectual Birdhouse. Artistic Practice as Research*. London, UK: Koenig Books, 2012, pp. 229–246.

[13] M. Puckette, "The Patcher," in *Proceedings of the International Computer Music Conference*, 1988.

[14] ——, "A case study in software for artists: Max/MSP and Pd," in *Art++*, hyx ed. David-Olivier Lartigaud, 2016.

[15] ——, "Combining Event and Signal Processing in the MAX Graphical Programming Environment," *Computer Music Journal*, vol. 15, no. 3, pp. 68–77, 1991.

[16] M. Csikszentmihalyi, *Creativity: Flow and the Psychology of Discovery and Invention*. New York: HarperCollinsPublishers, 1996.

[17] M. Weiser, "The Computer for the 21st Century," *Scientific american*, vol. 265, no. 3, pp. 94–105, 1991.

[18] B. Matuszewski and N. Schnell, "LFO - A Graph-based Modular Approach to the Processing of Data Streams," in *Proceedings of the 3rd Web Audio Conference*, London, UK, 2017.

[19] B. Ermentrout, "An adaptive model for synchrony in the firefly Pteroptyx malaccae," *Journal of Mathematical Biology*, vol. 29, no. 6, pp. 571–585, Jun. 1991.

[20] P. Brinkmann, C. McCormick, P. Kirn, M. Roth, R. Lawler, and H.-C. Steiner, "Embedding Pure Data with libpd," in *Proceedings of the Pure Data Convention*, 2011.

[21] A. McPherson and V. Zappi, "An environment for submillisecond-latency audio and sensor processing on BeagleBone Black," in *Audio Engineering Society Convention 138*. Audio Engineering Society, 2015.

[22] E. Berdahl, S. Salazar, and M. Borins, "Embedded Networking and Hardware-Accelerated Graphics with Satellite CCRMA." in *NIME*, 2013.