Meta-Xenakis

New Perspectives on Iannis Xenakis's Life, Work, and Legacies

Edited by Sharon Kanach and Peter Nelson

https://www.openbookpublishers.com

Sharon Kanach and Peter Nelson (eds), *Meta-Xenakis: New Perspectives on Iannis Xenakis's Life, Work, and Legacies*. Cambridge, UK: Open Book Publishers, 2024, https://doi.org/10.11647/OBP.0390

Digital material and resources associated with this volume are available at https://doi.org/10.11647/OBP.0390#resources

ISBN Paperback: 978-1-80511-224-2
ISBN Hardback: 978-1-80511-225-9
ISBN Digital (PDF): 978-1-80511-226-6
ISBN Digital ebook (epub): 978-1-80511-227-3
ISBN HTML: 978-1-80511-229-7
DOI: 10.11647/OBP.0390

Cover image: Iannis Xenakis at the C.R. MacIntosh Museum, Glasgow, Scotland, 1987. Photo by Henning Lohner, courtesy of CIX Archives, Lohner collection.

Cover design: Jeevanjot Kaur Nagpal

With generous support from: Centre Iannis Xenakis, Xenakis Project of the Americas/The Brook Center, Université de Rouen: BQRI, IRIHS, GRHis, CÉRÉdI.

40. The Xenakis Networked Performance Marathon 2022: An Experiment in Networked Performance Collaboration Inspired by Iannis Xenakis

Iannis Zannos, Martin Carlé, Vasilis Agiomyrgianakis, Takumi Ikeda, and Hanako Atake[1]

Introduction

This chapter documents the results of an international collaborative performance project organized in honor of Iannis Xenakis at the hundredth anniversary of his birth. Xenakis's work exerts lasting influence in the world of music as well as in the broader context of the avant-garde. His engagement with mathematics and technology from a broad philosophical scope within the program of modernism is a source of inspiration for artists working at the threshold between art and technology. Furthermore, his philosophical approach of music and the role of technology and mathematics in music forms a long-lasting foundation for creative engagement in the field.

In this context the "Xenakis Networked Performance Marathon 2022" (XNPM22) project was conceived as a challenge to engage with the legacy of Xenakis from the perspective of contemporary movements in music and performance arts, while also exploring the potential of online musical collaboration through the use of digital technologies falling within the extended scope of Xenakis's technological and

1 The authors would like to thank project HAL (Hub of Art Laboratories) at the Department of Audiovisual Arts Ionian University for the financial support which was indispensable for realizing the project. Thanks are equally due to Professor Haruka Hirayama of Hokkaido Information University for support and collaboration in the years leading up to the project, for her invaluable efforts to provide the organizational and technical framework for the trials leading up to the project, and for participating in the project both remotely and locally. Finally, the authors are indebted to Professor Satoru Takaku of Nihon Daigaku for providing valuable support and rehearsal infrastructure for this research.

 https://doi.org/10.11647/OBP.0390.42

mathematical vision: program code as an implementation of mathematical thought, and sensors as interfaces in experimental techniques extending the expressive modes of performance.

The project was realized through the collaboration of several institutions. The result was a marathon performance in the main Dance Studio classroom of Athens Conservatoire, presenting sixteen pieces from artists in ten different countries in the span of eight hours.[2] The marathon included four ensemble dance pieces using wireless wearable sensors tracking movement by means of inertia to control sound synthesis, as well as numerous pieces demonstrating networked performance with diverse software and approaches. The technical and organizational problems encountered are too many to report here in all detail.

Some of the solutions to these problems led to unexpected insights and to prototypes of software and hardware systems that merit further examination and development. Here we mainly center on the original work done for the development of alternative wireless wearable sensor hardware during the extensive rehearsals of the dance ensemble pieces. These can be seen as examples of frugal innovation, in the sense of assembling or modifying (hacking) low-cost hardware to meet the requirements of the project. We also outline the software framework used in most of the live coding performances and discuss the impact of self-contained minimal setups in other experimental performance practice forms such as a street performance that formed part of a complementary collaborative project in parallel to the marathon. In Chapter 41 in this volume, we cite the descriptions of all pieces as provided by the artists, in order to give an overview of the scope of the project, and provide links to the videos of the pieces.

IMU and Frugal Innovation in Dance-driven Music Performances: Overview

Inertial measurement units (IMU) are amongst the most ubiquitous and readily available sensors. We used IMU sensors to explore their potential as instruments for musical expression in a variety of settings that challenge and expand the conventional notions of (music or dance) performance, testing several different types of wearable IMU-based motion tracking devices, thus enabling dancers/musicians to control sound synthesis through their body movements. Hardware includes self-made devices based on Arduino, Linux OS, custom systems using Zigbee and N24 NRF protocols, off-the-shelf Wi-Fi devices such as iPhones or the ESP based M5Stick, low-cost health-monitoring watches using Bluetooth.

We encountered challenges at all levels: hardware, software and networking configuration, interaction design, and performance practice. A distinguishing

2 One piece needed to be cancelled due to illness of the performer, Thanos Polymeneas-Liontiris.

characteristic of the present work is that it required participation and collaboration between members of a small team, and that hardware, software, networking, sound design or compositional and choreographic work proceeded hand in hand in real time while developing the performances. This provided practical evidence that it is both possible and necessary to deal with these aspects in an integrated manner, and indicated the need for developing a work methodology that supports a close feedback loop between these components, and provides continuous creative motivation and guidance to artists.

IMUs and New Performance Practices

By combining readily available inertial measurement units with equally widely available wireless hardware components, one can build low-cost wearable systems for tracking human body movement. Sensor-based tracking of human movement is thus being established as a trend in live performance with body movement, parallel to camera-based systems. This opens up many possibilities, while at the same time requiring a fundamental rethinking of the methodologies and roles of the artists involved.

The ability of performers to trigger sounds or to modify the flow of sonic events by moving their bodies freely is a powerful and attractive feature. Yet it is not easy to build performances with these elements that keep the attention of the audience and build a captivating or meaningful narrative. Most importantly, the relationship between body movement and generated sound is complex and unexplored. Musicians/programmers and performers face the challenge of developing a language or system which interconnects body movement to sound in such a way that allows one to create performances that are satisfying and meaningful both to the artist and to the audience. This is a large task. The present chapter does not attempt to give answers but limits itself to reporting the experiences made through several projects and to provide information and ideas that may prove useful for future work in the field. We report as practitioners who develop fully fledged performances based on low-cost hardware. We believe that the best way to develop the new languages hinted at above is to collaborate on hardware, software, and performance design methodology, and we therefore deal with all these aspects in this chapter.

Musician/Programmer and Dancer: Form Design, Improvisation, Interaction Design

While the use of motion sensors in music technology is gradually becoming widespread, most of the basic questions regarding their aesthetic function and their nature as a form of performance remain unanswered. Why employ motion sensors instead of other controllers? What is the intention of including movement in music in the first place? Dancers are arguably more aware than musicians about how to

express themselves by moving their bodies. However, they are less accustomed than musicians to having their movements directly converted into sound. Even if a musician or programmer, assigns different functions to each of the three motion axes of a sensor for musical reasons, it is very unlikely that the dancer will move as the musician or programmer intended. For this reason, we adopted an alternative methodology, which requires the musician to consider the characteristics of dancer's movements when programming the sensors. The dancer performs with the sensors while also taking their choreography into account. Then the musician adjusts the program according to the dancer's choreography. The performance is gradually polished over iterations of these processes. In light of the above, describing this piece simply as music or dance is inappropriate. The present project thus ventures into the field of performances that do not belong to an established genre. These represent at the same time a threat to prevailing conventions and a way to escape the system. We view this is a first step towards defining performance conditions and paradigms better suited for performance with motion sensors. Here we focus on two performance categories that venture outside the conventional concert-hall configuration: (a) Telematic performances where the performers play in remotely located venues at the same time and coordinate through motion data and code sent over the internet and (b) mobile performances in open public spaces. These settings present concrete technical and methodological challenges which we discuss below.

Background and Challenges

The initial goal of this project was to investigate how dancers can perform in remote locations simultaneously by sharing their body movement through data over the internet and creating sound from this data in real time. Sound is generated by SuperCollider from data shared via Open Sound Control (OSC).

Since 2018, several performances based on this idea have been realized in different settings, including telematic performances between up to three different locations and local performances. Challenges arising from this task are:

1. Capturing the data with sensors and sending it to the music-making workstations.
2. Transmitting these data over the internet, across firewalls and to computers connected to local subnetworks.
3. Designing interaction algorithms to produce sound (and graphics) from the data.
4. Adapting the algorithms to be used by the dancers. This presupposes a common understanding between dancer and programmer/musician about the principles governing body movement and sound and the relationship between these two.

5. Designing performances together with the dancers, resulting in choreographies coupled with music.

To the above should be added the overall challenge of working with multiple hardware and software, and ideally also different interaction strategies, at the same time. This is an essential requirement to permit collaboration, and therefore a necessary condition to develop a shared performance practice. The following sections describe the work of the project in these fields.

Performance Criteria for Wireless Communication Standards

We explored different wireless protocols: Wi-Fi, Bluetooth, Zigbee, and Nordic Radio Frequency wireless data protocol (NRF24). The criteria which emerged from work in different performance settings are:

1. Range and reliability: How far can the sensor be located from the receiver and still transmit data? Can the sensor transmit data even when there are other physical objects between the sensor and the receiver? These characteristics depend on whether the sensor is in a closed space in which reflections from other surfaces amplify or re-transmit its signal, and on other characteristics such as the permeability of objects interjected between the sensor and transmitter, and the quantity of wave absorbing surfaces in the room. When a sensor loses its connection to the transmitter, how quickly and reliably can it reconnect? Reconnecting depends mainly on the software that manages the connection, and which may require fixing in order to reconnect faster.
2. Power requirements: How much electrical power is required to connect the device to the network? Implementation standards that use much power may produce a stronger signal and achieve better range. However, requiring less power is preferable, because it allows devices to work for a longer time with the same battery capacity.
3. Number of devices that can be connected to the network at the same time. Zigbee supports up to 256 devices in a Network, while Bluetooth supports up to seven and RF24 up to six devices per receiver.
4. Cost and availability of individual sender and receiver modules. These may vary according to time and country. Consequently, the information that we give here is bound to the local conditions in Greece during 2020–2. The cost factor can become irrelevant in certain situations when using devices that are already owned by the performers, such as smartphones.
5. Dependency on transmitter hardware and existing internet infrastructure: using Wi-Fi for sensor data transmission can interfere with the connection

of the main workstation to the internet which is required for telematic performances. This disadvantage, as well as the difficulty of setting up a connection in some situations, speaks against using Wi-Fi for wireless IMU performance.[3]

6. Robustness, size and ease of use of the device. How long does the device function on a single battery charge? How easy is it to recharge and to wear? How resistant is it to shock and fast movements?

Early Attempts

Our first prototype was made by connecting a Bosh BMF055 9 axis IMU to a CHIP Pro minicomputer running Alpine Linux.[4] Support for this system was discontinued at early stages of our work in 2018 due to failure of the manufacturing company. Subsequently, we continued with an Arduino-compatible Feather Huzzah ESP module by Adafruit, and finally to a Raspberry Pi Zero. Because there was no working Arduino library for Bosh BMF055, we substituted it by an Adafruit LSM6DSOX 6 DoF Accelerometer and Gyroscope. With this system, we were able to do several performances. However, connecting to Wi-Fi proved unreliable in many situations, even when using our own stand-alone Wi-Fi router. It furthermore required custom reconfiguration of routers when we wanted to connect to remote locations on the internet, which were not always feasible. On the whole, our experience confirmed the arguments already presented by other researchers that it is strongly preferable to use a dedicated (Wi-Fi-independent) protocol.[5]

Sense/Stage System (Zigbee)

Sense/stage is a system developed by Marije Baalman specifically for use in interactive performance.[6] It uses the Zigbee protocol. The data are received by the computer via a communication module that connects via USB. Its independence from Wi-Fi is a major advantage. The system performs well in open spaces at distances up to fifteen meters at direct line of sight, and in medium-sized closed spaces with few performers. However, when rehearsing in larger spaces with greater numbers of performers, we noticed that data transmission tended to diminish in rate or to stop, especially when the performer's body lies between the sensor and the receiver. For these performances, we sought alternative solutions, as shown next.

3 Baalman et al., 2017.
4 Zannos and Carlé, 2018.
5 Baalman, 2017.
6 Baalman et al., 2009.

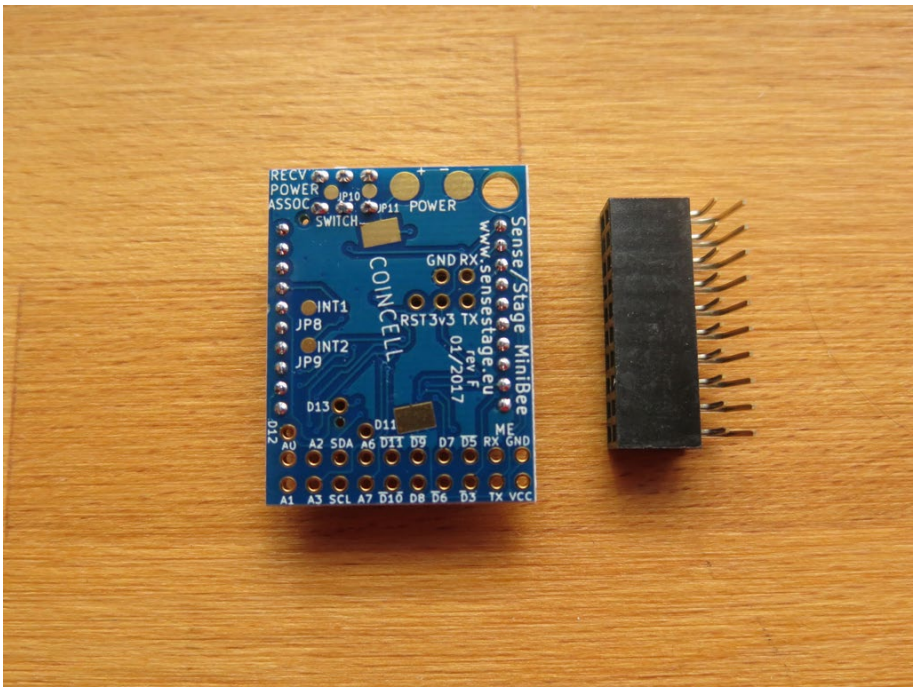

Fig. 40.1 Sense/Stage MiniBee with Header. Photo by Iannis Zannos (2023).

Smart Watches Using Bluetooth

As replacement for the XBee-based sense/stage system we use a wireless sensor system built on low-cost, nRF52-based smart watches providing real-time accelerometer data as OSC messages. Two smart watch models outwardly similar in hardware but otherwise very different were employed: the PineTime smart watch ships with an open-source firmware and OS by PINE64.[7] Variants are available under various names by a no-name Chinese manufacturer and use proprietary technology (P8). Lack of time and financial resources and limited market availability forced us to use both watch models. We canvassed through small low-tech shops in Athens selling out their remaining stock of "Colmi P8," a medium-priced health tracker popular in Southern Europe two years ago, and then hacked the gleaned models to access the data. While for our purposes we focused on accelerometer data, these watches offer many other options for feedback and control which are accessible through a programmable touchscreen, which invite creative uses (see Figure 40.2).

7 "PineTime" (13 January 2024), *Wiki.pine*, https://wiki.pine64.org/index.php/PineTime

Fig. 40.2 PineTime Smartwatch showing X-Y-Z Data. Photo by Martin Carlé (2023).

The smart watches are compact sealed devices designed for athletic workouts. This makes them ideally suited for use in dance performance because they can be easily attached in various places on the body and are shock and moisture resistant. The thin wire aerial inside these watches (see Figure 40.3) connects to the integrated Bluetooth Low-Energy radio of the Nordic nRF52832 chip[8] providing wide signal range and high data throughput.

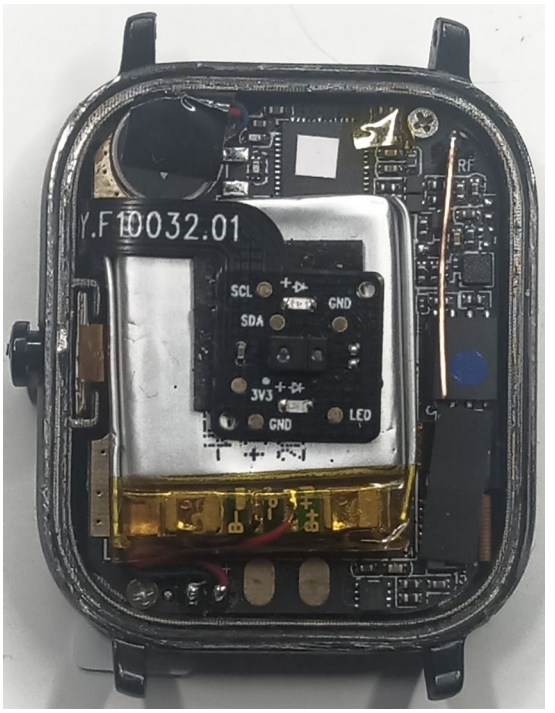

Fig. 40.3 P8 opened showing thin wire aerial. Photo by Martin Carlé (2023).

8 Nordic, 2022.

When paired with a class 2 wireless chipset of a common laptop, PineTime outperformed the XBee S1 [8] radio module of the Sense/Stage MiniBee [9] adapter. However, to achieve the required range of fifteen meters, an omni-directional antenna attached to an external BLE 5.0 class 1 adapter was necessary.

Improving The Fault Tolerance

While the BLE connection had a stable update rate, in would entirely stop transmitting when the signal strength dropped below a critical threshold. In these cases, the open-source software for BLE on Linux took several seconds to re-establish a connection to the smart watches. In order to overcome this disadvantage, we had to find a way to reduce the recovery time to a minimum. This required us to search amongst a variety of environments and software platforms for a solution. We summarize the required work in the following section.

Accessing PineTime Software and Firmware

Since its launch in 2020, PineTime has been widely used as a development platform. Due to its open nature and the widespread use of the Nordic nRF52 series for Internet of Things (IoT) projects there is a variety of real-time operating systems available, such as Zephyr or RIOT, and a number of language specific runtimes, from RTFM (Rust), and Tiny Go, to wasp-os (MicroPython).[9] PineTime ships with InfiniTime an OS written in C++ that builds on FreeRTOS and employs the NimBLE open-source Bluetooth 5.1 stack as a complete replacement for the proprietary SoftDevice on the Nordic chipsets \cite{nimble}.[10] We had to modify these environments in order to modify the software responsible for the recovery of BLE connections. InfiniTime supports Over The Air (OTA) firmware updates and even has a recovery firmware in place. Yet a sealed PineTime cannot be used for development because it may brick the device. We encountered this when attempting an OTA hack of the P8. To bypass this obstacle, we employed the DaFlasher software by Aaron Christophel.[11] This software exploits the stock DaFit firmware to install a custom bootloader which in turn allows a downgrade of the Nordic SoftDevice to an unprotected version, enabling the use of several other flashing tools for hacking the device. At present, this involves a daunting series of nine delicate steps, continuously replacing one bootloader after the other with wasp-os as intermediary, and finally enabling InifiniTime on the P8. If something goes wrong, the device is bricked, i.e. inaccessible, and one needs to perform a wired

9 "PineTime Development" (21 April 2023), *Wiki.pine*, https://wiki.pine64.org/wiki/PineTime_Development

10 InfiniTimeOrg, "InfiniTime" (2022), https://github.com/InfiniTimeOrg/InfiniTime; apache, "mynewt-nimble" (2022), *GitHub*, https://github.com/apache/mynewt-nimble

11 Aaron Christophel, "DaFlasherFiles" (2022) *GitHub*, https://github.com/atc1441/DaFlasherFiles

hardware glitch-hack using Christophel's ESP32 SWD Flasher.[12] Eventually, the seal had to be broken on both the PineTime and the P8. Each firmware update had to be tested on each of these development devices before installing it on the production watches.

Final Steps

Of the various methods for accessing the IMU of PineTime or P8, InfiniTime OS offers a uniform and nearly interruption-free method for accessing motion data in real-time. In addition to companion apps, like Gadgetbridge or AmazFish, which use standard BLE implementations on both mobile and desktop platforms, InfiniTime also supports a custom motion service.[13] This service can be accessed through any BLE API (application programming interface). On Linux one can use the daemon application itd implemented in Golang by Arsen Musaleyan to forward motion data received from BLE to a standard UNIX socket.[14] We used the liblo [19] and GoOSC libraries to forward these to the operating system via UDP as OSC messages.[15]

Finally, two more hacks were necessary: first, we needed to alter ltd so that it could handle multiple watches at the same time, and secondly, we had to modify the source code of the Goroutines in GoOSC to allow for quick recovery from data connection failures, thereby reducing the amount of offline time to an imperceptible minimum. After resolving those problems, the restriction of having a maximum of seven devices per master adapter via Bluetooth was easily overcome by using additional low-cost BLE adapters. In our experiments we used up to twelve IMU smart watches connected via Bluetooth at the same time. If more devices were available, we could have increased this number easily.

Comparing Sense/Stage with Pinetime

When comparing connectivity beyond a distance of six or seven meters, we found the sense/stage modules had a drastically reduced data update frequency, whereas PineTime and the P8 continued to send motion data at the same rate. This was especially conspicuous in stage settings with weak radio reflections (compared to rehearsal spaces), or in choreographies where bodies frequently twist around or line up creating a barrier between the sensors and the receivers. In such situations, the actual transmission distance may reach only about a fifth of the thirty meter range indicated by the Zigbee protocol documentation.[16] Since in our performances the

12 Aaron Christophel, "Swd Flasher Files" (2022), *GitHub*, http://atcnetz.blogspot.com/2021/06/ esp32-swd-flasher-fur-die-nrf52-serie
13 InfiniTimeOrg, "InfiniTime" (2022), https://github.com/InfiniTimeOrg/InfiniTime
14 Elara6331, "itd" (2022), https://gitea.arsenm.dev/Arsen6331/itd
15 Hypebeast, "go-osc" (2022), *GitHub*, https://github.com/hypebeast/go-osc
16 "Zigbee Protocol," *ScienceDirect*, https://www.sciencedirect.com/topics/engineering/zigbee-protocol

sensor data were directly mapped to audio synthesis parameters, it was critical to maintain a constant update rate. Under these conditions the PineTime and P8 smart watches clearly outperformed the sense/stage system. As a result, for the conditions set by our performances (continued long range under low-reflectivity conditions and large number of performers) the BLE based PineTime watches were preferable compared to sense/stage.

NRF24-Based Prototype

In parallel with the development of the PineTime based devices, we also started work on an alternative system using NRF24 transmitters coupled with Arduino nano. NRF24 is a radio frequency wireless protocol with excellent long-range performance. Its transmitter and receiver modules are available at very low prices, as shown in the list at the end of the present section. This solution relies on transmission technology with low-power radio, using an Arduino-programmable microprocessor (Arduino Nano) with a wireless transceiver module (nRF24L01) and a 6-DOF motion sensor (MPU6050). Despite the reservations due to limited number of nodes per adapter, expressed by Baalman, we thought it worthwhile to evaluate this technology because it promises longer range and stabler connectivity.[17]

The radio transceiver module nRF24L01 is a low power and low-cost wireless module by Nordic using a 2.4G license-free ISM band. According to the specs, this module can reach a range of one hundred meters with its built-in antenna and up to one kilometer using an external antenna. One receiver can receive data from multiple transmitters. Both transmitters and receiver use the same wireless Module (nRF24L01). Figure 40.4 shows the prototypes in development.

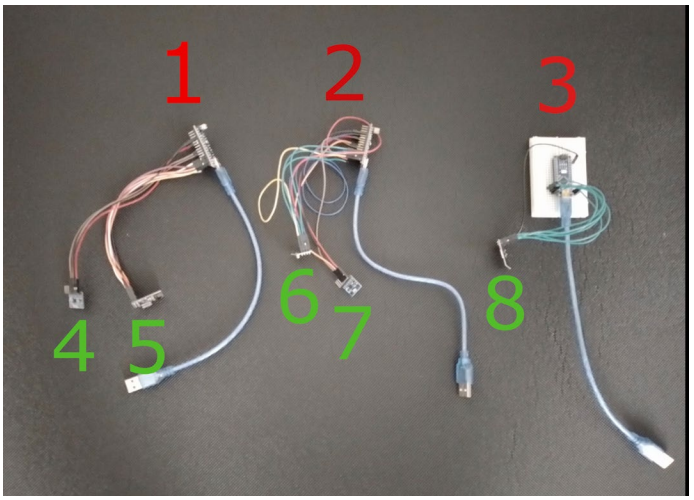

Fig. 40.4 nRF23L01 prototype: (1, 2) transmitter, (3) receiver, (4, 7) sensor, (5, 6, 8) transceiver. Photo by Vasilis Agiomyrgianakis (2023).

17 Baalman et al., 2017.

The transmitters are connected with a battery via USB. We added a DC-DC step up converter booster power supply module for connecting 3.7V 1000mAh Lithium Polymer rechargeable batteries.

We employed a six-axis accelerometer-gyroscope sensor (MPU6050) and used the x, y, and z axes of the accelerometer to measure movement data. We tested this system in rehearsals with actors and dancers at Athens Conservatoire and in Hokkaido, Japan in preparation for the Xenakis Networked Performance Marathon.

Programming in Arduino IDE is the first step in using this system. We programed the Arduino microcontroller with the nRF24L01 (transmitters and receiver) to communicate at the maximum distance that the particular chip offers (one hundred meters with its built-in antenna) as well as behind surfaces like walls, dancers' bodies, and other obstacles. The data was then sent serially to the computer via USB, using Python 3 to relay the data to SuperCollider in the form of OSC messages. To adjust sound characteristics like pitch, loudness, grain density, and duration in the granular sound synthesis technique among others, we use individual parameters from the accelerometer axes (x, y, and z). Additionally, we designed the circuit layout in preparation for a stand-alone, wearable system. Figure 40.5 shows the Arduino system connected to the IMU sensor and the N24L radio transmitter.

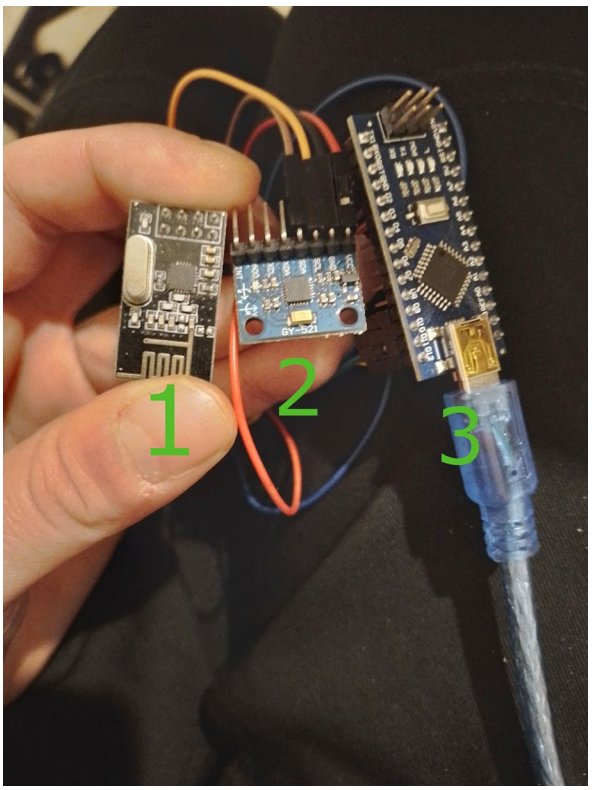

Fig. 40.5 The transmitter with (1) the nRF24L01 transceiver module, (2) the MPU6050 IMU sensor and (3) the Arduino nano. Photo by Vasilis Agiomyrgianakis (2023).

This system is still under development. The prototypes performed with constant update rates even when the sending module was behind a wall or a number of human bodies.

Software

Sensor-based dance performances involving concurrent remote locations encounter the problem of exponentially growing complexity of control parameters and coordination commands. The present project addressed this issue by sharing all data and all code between the performing locations in real time. In that way, the performers at each venue are participating in one virtual performance, consisting of the shared data and control code. This virtual performance is physically translated into replicas at each location. Thus, by definition, the performers and the audience share a common experience independently of the actual physical space in which they are located. The data sharing is implemented through the use of the open-source software OscGroups by Ross Bencina.

Increasingly complex challenges arise when one tries to use this system in settings that are not limited to a single user, type of sensor hardware, or rendering software system. Thus, the software architecture must be designed in order to be capable of operating under the following three conditions:

1. Multiple live coders performing at the same time.
2. Multiple different types of sensors with different signal specifications and OSC message protocols are in use in the same system.
3. Multiple different types of software for sound and/or graphics are operating at the same time in one performance setting.

We outline here the design principles that resulted from this research, and which are implemented in the SuperCollider library found on GitHub.[18]

1. Using Multiple Environment Dictionaries

This principle follows one of the fundamental design patterns of JITLib, an early and widely used environment for live coding in SuperCollider.[19] This is the idea of a ProxySpace, that is an environment where objects are stored as dictionary entries, and accessible to the user as environment variables when the ProxySpace is made current by the command "push." The present library defines a subclass of EnvironmentRedirect, called Mediator which is responsible for storing sound processes and the objects

18 See Iannis Zannos, "sc-hacks-redux," *GitHub*, https://github.com/iani/sc-hacks-redux. An early version is discussed in Zannos, 2018.

19 Rohrhuber et al., 2005.

associated with them such as control busses, but which behaves in a different way than ProxySpace in JITLib, as explained below.

2. Stop Processes When Replacing Them

One of the first obstacles faced by live coders in SuperCollider is keeping track of sound processes stored in variables. When such a process is replaced by a new one, the reference to the old one can be lost, resulting in a hanging sound which cannot be stopped. To help with this, Mediator automatically stops a sound process when something else is stored in its place.

3. Operators for Creating and Modifying Sound Processes

The library operators such as +> and ++> are used as shortcuts for commands that convert a function or a dictionary of type Event into a sound signal synthesis process or a stream generating many sound events, and for modifying the state of sound syntheses or event streams.

4. Operators for Accessing Control Busses and Buffers

Sensor-based systems for music performance commonly create a mapping between the values of the parameters input from the sensors into parameter changes of the sound synthesis processes. This rapidly becomes complicated when many sensors with many parameters of possibly different value ranges are involved, because each of these must be mapped into parameter changes in possibly many different parameters of different sound synthesis processes. Additionally, mapping strategies are not just limited to linear or exponential scaling between value ranges, but may also involve logical decisions denoting state changes such as on or off, or translation into trigger events that initiate event state changes (start or stop a sound event). It is therefore necessary to separate processes that operate on signals to produce control signals or triggers from processes that produce sound in response to control signals. For this reason, the sc-hacks-redux library of this project defines operators and shortcuts for accessing control busses, mainly denoted by use of the @ mark (for example @>), and for defining signal processing processes that operate on these. As a result of this distinction, it is possible to separate the design of control signal processing from the design of sound signal synthesis, and to combine these by using as keys the names of busses pointing to control signals. One future potential of this approach to be explored is the possibility to design information processing networks in a manner resembling artificial neural networks.

5. Use of Observer Pattern

The Observer pattern is a widely used language pattern in object-oriented programming. It is based on the principle of a notifier object issuing a notification that it has changed, which is then received by any other objects that may need to respond to the changes in the notifier. The library of this project defines a new class Notification that implements this pattern along with an API that enables one to easily define custom responses of any object to the message changed when received from another object. This pattern is almost ubiquitously used in the library and serves as basis for designing both graphical user interfaces (GUIs) and for customizing the behavior of the system in response to OSC messages. The Notification class combines two mechanisms implemented in the built-in SuperCollider library with the methods update and changed and with the class NotificationCenter, in such a way as to provide a uniform solution for all Observer pattern use contexts. In this way it provides a basis for programming asynchronous communication in both GUI and server-language object updates.

6. Dynamic Definition of Responses to OSC Messages

Based on the Observer pattern protocol of the Notification class, the library defines operators and methods which enable the user to modify the response of the language to incoming OSC messages on the fly. Selected examples of functionalities enabled by this feature set of the sc-hacks-redux library are:

- Automatically send code strings locally evaluated by the user to other users in the network, using OscGroups.
- Evaluate locally code strings received from other users via OscGroups
- Write numeric values received via OSC messages into control busses

This results in a tool that enables coders to customize the behavior of the system and is therefore the current focus of the present research, along with information processing networks.

7. Code Management

Making use of the basic tools mentioned above, the library provides a basic GUI framework for navigating a folder system holding SuperCollider code files, and for executing any of those files either as a whole, or in part based on the separator `//:`. Furthermore, it is possible to play any such file as a score where each part or code snippet has a predefined duration given in the comment header in the form of `//:[duration]`. This makes it possible to write scores as program files and to play these at any moment from the GUI.

8. Saving and Replaying of OSC Data

Since OSC is the de facto basis of communication both between sensor hardware and between live coders in a performance, the library provides a way to store all data in any performance session into a series of files as SuperCollider code and to replay these later for experimentation purposes.

Performance Practice

1. Networked Performance

Our telematic performance settings are similar in principle to those encountered by online game players sharing an experience in a virtual world, where the actions of each player are always transmitted to and reflected in the local systems of all other players, thereby creating a virtual space or game-universe shared by all players independently of their locations or gaming platform hardware. The performance is controlled by live coding in SuperCollider where the coders are responsible for configuring, mapping, and routing the sensor input from the dancers to the various rendering modules. In practice, it is possible for a single user to run a performance remotely with multiple dancers performing on multiple remote locations. The major difference, however, is that the players are controlling the sound by performing on stage with their entire bodies. This creates an entirely new situation, whose analysis lies beyond the scope of the present chapter.

2. Outdoor Performance

In some cultures, plazas are not inherently accepted as public spaces for performance, so alternative spaces are used, such as public roads. However, such locations serve only as temporary performance spaces, and attending audiences eventually disband. In societies that do not support plaza culture style, arts are usually allowed to exist only in closed spaces such as theaters, concert halls, and galleries whose function is predefined. However, it can be argued that the major factor inhibiting street performances is the influence of cultural stereotypes, rather than institutions themselves. A prevailing stereotype underlying the performance paradigm in closed spaces is the presupposition of a sequestration of the senses to allow the audience to focus their undisturbed attention on the performance. By contrast, performances in open spaces receive interference from various aural and visual sources, such as passersby and unintended sounds perceived as “noise.” Therefore, the performance must creatively embrace the constraints arising from the circumstances in which it is embedded. The performer is obliged to function within these constraints. We experimented with performances that expose both the dancer and the live coder jointly

to these conditions and set them the task to face together, on the spot, the challenge of creating new performance constraints outside the box of predefined computer-music performance settings (see Figure 40.6).

Fig. 40.6 Street performance with M5Stick and portable Wi-Fi system. Photo by authors.

Such a performance setting served as testing ground for working with one more IMU system. We used M5Stick, a thumb-sized off-the shelf sensor by M5Stack, equipped with gyroscope and accelerometer sensors. This uses ESP-based Wi-Fi. To make our system mobile, we employed a compact USB-powered Wi-Fi router by TP-Link (TL-WR802N Wi-Fi Nano) (see Figure 40.7). Sound was generated by SuperCollider on a laptop and rendered via battery-powered speakers. This gave us full self-sufficiency and mobility. The internet connection was hard coded in Arduino on the M5Stick, rendering connection immediate and automatic. This system served well in various outdoor urban locations. We recorded gyroscope and accelerometer data for comparison with that of other systems.

Fig. 40.7 Mobile system with M5Stick and mini-Wi-Fi router. Photo by authors.

References

BAALMAN, Marije (2017), "Wireless Sensing for Artistic Applications: A Reflection on Sense/stage to Motivate the Design of the Next Stage," in *Proceedings of the 17th International Conference on New Interfaces for Musical Expression* (*NIME 2017*), Aalborg University, Copenhagen, p. 307–12.

BAALMAN, Marije, DE BELLEVAL, Vincent, SALTER, Christopher, MALLOCH, Joseph, THIBODEAU, Joseph, and WANDERLEY, Marcelo (2009), "Sense/stage - Low Cost, Open Source Wireless Sensor Infrastructure for Live Performance and Interactive Real-time Environments," in *Proceedings of the 7th International Conference on New Interfaces for Musical Expression* (*NIME 2007*), Pittsburgh, Pennsylvania, p. 131–4.

NORDIC (2022), *Nordic nrf52832 Documentation*, https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.4.pdf

ROHRHUBER, Julian, DE CAMPO, Alberto, and WIESER, Renate (2005), "Algorithms Today: Notes on Language Design for Just in Time Programming," in *Proceedings of the International Computer Music Conference*, Barcelona, ICMC, p. 455–8.

XENAKIS, Iannis (1992), *Formalized Music: Thought and Mathematics in Music*, additional material compiled and edited by Sharon Kanach (rev. ed.), Stuyvesant, New York, Pendragon.

XENAKIS, Iannis (2008), *Music and Architecture: Architectural Projects, Texts, and Realizations*, compilation, translation, and commentary by Sharon Kanach, Hillsdale, New York, Pendragon.

ZANNOS, Iannis (2019), "sc-hacks: A Live Coding Framework for Gestural Performance and Electronic Music," in *Proceedings of the 2019 Linux Audio Conference*, Stanford University, California, CCRMA, p. 121–8, https://lac.linuxaudio.org/2019/doc/zannos.pdf

ZANNOS, Iannis and CARLÉ, Martin (2018), "Metric Interweaving in Networked Danced Music Performance," in A. Georgaki and A. Andrepoulou, *Proceedings of the Sound and Music Computing Conference 2018*, Limassol, Cyprus, https://ec.europa.eu/programmes/creative-europe/project-result-content/d6fe25c8-afc8-405e-999c-3853298711b8/IU_Metric-Interweaving-for-Dance-and-Music_Zannos_SMC2018.pdf