The Puara Framework: Hiding complexity and modularity for reproducibility and usability in NIMEs

Eduardo A. L. Meneses
SAT/Metalab
Montreal, Canada
emeneses@sat.qc.ca
eduardo@edumeneses.com

Thomas Piquet
SAT/Metalab
Montreal, Canada
tpiquet@sat.qc.ca

Jason Noble
Université de Montréal
Montreal, Canada
jason.noble@umontreal.ca

Marcelo M. Wanderley
IDMIL, CIRMMT
McGill University
Montreal, Canada
marcelo.wanderley@mcgill.ca

ABSTRACT

This paper presents Puara, a framework created to tackle problems commonly associated with instrument design, immersive environments, and prototyping. We discuss how exploring Digital Musical Instruments (DMIs) in a collaborative environment led to generalizing procedures that constitute a starting point to solve technical challenges when building, maintaining, and performing with instruments. These challenges guided the framework organization and focus on maintainability, integrability, and modularity. Puara was employed in self-contained systems using 3 DMI building blocks (network manager, gestural descriptors, Media Processing Unit) and supporting 3 established DMIs (GuitarAMI, T-Stick, Probatio) and one new instrument (AMIWrist). We validated Puara with two use cases where parts of the framework were used. Finally, we accessed the influence of frameworks when exploring predefined NIMEs without concern about the inner workings, or shifting composition paradigms between event-based and gesture-based approaches.

Author Keywords

Music-related human-computer interaction; Reproducibility, Hiding complexity; instrument design frameworks

CCS Concepts

•**Applied computing** → **Sound and music computing; Performing arts;** *Media arts;* •**Computer systems organization** → **Embedded systems;**

1. INTRODUCTION

Researchers and artists have increasingly explored gestural controllers for music applications since the 1980s. The popularization of the MIDI protocol and the availability of low-cost computing devices are cited as reasons electronic musical instruments became ubiquitous [17]. We can also associate availability and low cost with the emergence of diverse tools for media arts, including Digital and Augmented Musical Instruments (DMIs/AMIs), devices and systems for art installations, and interactive environments.

The exploration of these new tools raised the technical requirements and skills needed to design, build, reproduce, compose with, and perform with New Interfaces for Musical Expressions (NIMEs), or Internet of Musical Things [22]. Digital instrument designers and builders need extensive knowledge of electronics and manufacturing. Sensors, signal conditioning, filtering, soldering skills, and Computer-Aided Design (CAD) are often used to create NIMEs. Performers and composers are often required to have expertise in communication protocols to transmit and receive data between gestural controllers and sound synthesis units (and networking protocols in some cases). Music programming languages are also required tools for anyone willing to use digital instruments.

Although all the required knowledge is valuable, it is unfortunately not part of most music academic curricula. The impact of these new expertise requirements can be seen when artists try to meet the high level of demand in artistic activities [2], and the limitations of engineering solutions often found in digital instrument design [13]. Moreover, there are still several challenges when working with new media art and instrument design from the music technology perspective. Some of the building challenges we tackled during this research include: 1) building and design, 2) maintainability and entry fee, and 3) managing complexity. We will discuss the implications of each item in Section 3.

The exploration of these challenges was triggered by the design and reproduction of three DMIs/AMIs developed at Input Devices and Music Interaction Laboratory (IDMIL): the T-Stick [18], the GuitarAMI [15], and Probatio [4].

2. EXPLORED DMIs

2.1 The T-Stick

The T-Stick is a gestural controller initially created at IDMIL and the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT) in 2006. The current version of the T-Stick has capacitive sensors, an inertial measurement unit (IMU), a piezoelectric transducer, a force-sensing resistor (FSR), and, in some versions, infrared, air pressure, and light sensors. The controller is usually built using a

NIME'23, 31 May–2 June, 2023, Mexico City, Mexico.

cylindrical tube.

The work described by Nieva et. al. [18] paved the ground for students to build T-Sticks in a discipline entitled *Gestural Control of Sound Synthesis*[1], taught at McGill University in 2019. The discipline includes a practical assignment where students build DMIs and gestural controllers or perform experiments using these devices. The building guide provided to the students (accessible at `http://www.idmil.org/education/mumt620-t-stick/`) by Nieva presented step-by-step instructions on building the controller and uploading the firmware, including images and complete descriptions to guide the student through the process. Graduate students from the music technology area solve design problems and update the building instruction manual to allow the students enrolled in the discipline to fulfill the assignment. One example of this "on-demand" problem-solving is a redesign to allow replicating the T-Stick without cutting the pipe, as we can see in the updated building instructions at the most recent building instructions[2].

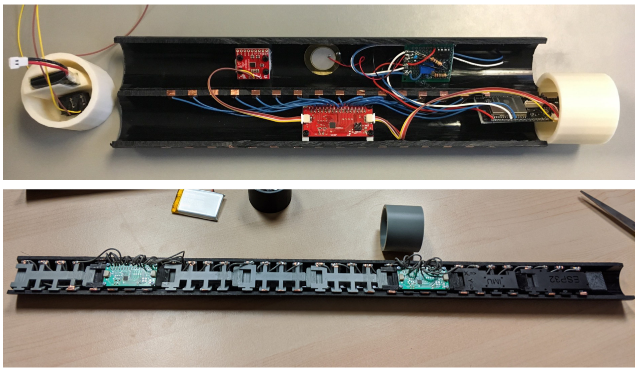

Figure 1: A sopranino T-Stick that was built using 3D-printed supports (beds) under sensors (top image), and a soprano T-Stick that was built using a modular 3D-printed frame (bottom). It became unnecessary to cut the PVC tube lengthwise for the latter building process (the cut tube was used in the image for illustrative purposes) [14].

Along with hardware, strategies for maintaining software and providing easy instructions to upload firmware to the T-Stick were also essential parts of the design process. Converting sensor fusion algorithms and data processing into standalone libraries became crucial to maintaining the T-Stick code and allowing students to work on specific aspects of the instrument independently.

2.2 GuitarAMI

The GuitarAMI is an augmented instrument based on the classical guitar. The device was first designed and built in 2014 using an ultrasonic sensor, an IMU, and a laptop responsible for audio manipulation and feature extraction. The motivation for building the GuitarAMI was to overcome some of the classical guitar's intrinsic sonic limitations, such as short sustain and the lack of sound intensity control after the attack [14].

Different versions were built as the GuitarAMI was actively being developed between 2016 and 2018 at IDMIL. Even though these versions share similar hardware, the Sound Process Unit (SPU) uses different software and music programming language depending on the version. Version 4 is based on Prynth [6] and runs SuperCollider[3] code, while versions 5 and 5a use a modified Linux distribution capable of running SuperCollider code, Pure Data patches, audio plugins in LV2 format, and custom software installed by the user.

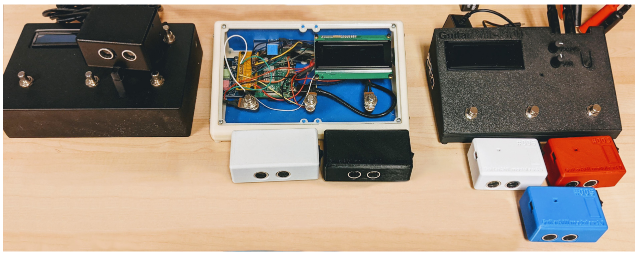

Figure 2: GuitarAMI versions and their modules: version 4 (left), version 5 (center), and version 5a (right). Version 4 is based on Prynth, while versions 5 and 5a use a custom Linux distribution.

The development of GuitarAMI's SPU provided tools to establish methods for connecting data from gestural controllers and synthesis processes. In addition, experiments performed with different frameworks for DMIs and AMIs during GuitarAMI-related projects [16] ensured flexibility to use other controllers and synthesis units interchangeably.

2.3 Probatio

Probatio is a toolkit for prototyping DMIs [4]. It consists of blocks, bases, hubs, and supports that can be easily combined to create functional prototypes that can be played in different positions and postures. The current prototype (version 1.0) was developed in an international partnership between research laboratories in Brazil and Canada. Probatio has been used in artistic performances (in Brazil) and research in music technology, more specifically, instrument design.

Similar to the T-Stick and the GuitarAMI, Probatio's multiple prototypes and different building processes add complexity to hardware maintainability. In addition, Probatio blocks can be rearranged, allowing users to prototype functional digital instruments in real time. Such flexibility requires complex map management between sensor data and sound synthesis parameters. Using and embedding mapping tools such as libmapper, or creating methods for the user to set the destination of Open Sound Control (OSC) messages in real time help fulfill these requirements.

3. BUILDING CHALLENGES

3.1 Designing and Building

For designing instruments and art installations, DMIs/AMIs can employ different combinations of controllers and synthesis units. Those parts can be built using different sensors, microcontrollers, CPUs, etc. Those devices also vary significantly according to the purpose, e.g., research, music performance, or immersive spaces and art installations.

In addition, researching a particular aspect of art and performance (including the effects of music/visuals on participants) requires data to be retrieved/archived. The devices also need to be consistent in their output if the research question involves comparing interaction between gestural controllers and different participants. Examples of this kind of study include: how different artists perform similar gestures [7], how researchers evaluate user experience for NIME

[1] `https://www.mcgill.ca/study/2019-2020/courses/mumt-620`

[2] `https://github.com/IDMIL/T-Stick/blob/master/Docs/T-Stick_2GW_building_instructions(trill).md`

[3] `https://supercollider.github.io/`

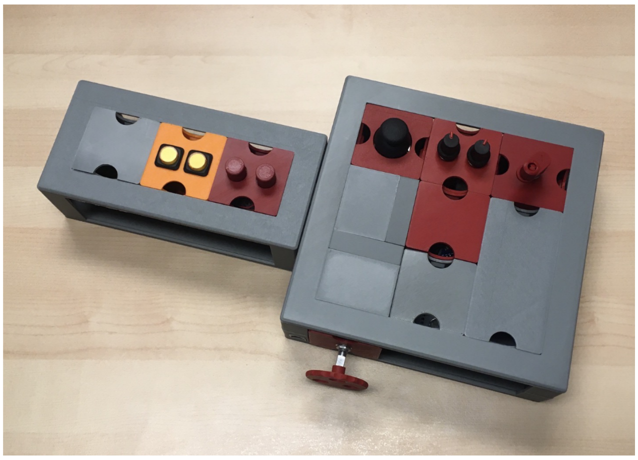

Figure 3: Probatio assembled with color-coded blocks. Red blocks output continuous sensor data, orange blocks output discrete sensor data, and grey blocks don't output data but provide structural support.

[20], the biological response or impact of musical stimulus [24], or DMI and mapping exploration [8].

Instrument designers creatively search for new interaction models and paradigms, often requiring consistent sensor data processing, robustness, and reliability [21].

Immersive spaces or art installations often require reliability and different levels of interaction between space, audience/visitors, and the material artistically explored [25]. The series of immersive experiences[4] and artistic residencies[5] conducted at the Société des Arts Technologiques (SAT) during 2022 highlight the level of reliability required for those artistic events and explorations.

3.2 Maintainability and entry fee

For maintaining instruments, we have, in addition to hardware replicability [3], a software layer, usually in the form of firmware, to make the hardware work. Maintaining code in a research-oriented environment where the creator is often a student that will eventually move on to another project or topic is difficult. The lack of team continuity is particularly detrimental to instrument longevity.

As users, artists often lack the skills to troubleshoot technical problems as they were trained to make art, not computer science or information technology. Wessel and Wright refer to interactions between users and devices (and mappings) when discussing *low entry fee* [23] for Musical Control of Computers. We can also extrapolate this idea to setting up and technically interacting with devices. Projects such as the T-Stick Music Creation Project [10] and the already mentioned artistic residencies at SAT/Metalab highlight the difficulties artists encounter even before receiving any gestural data or generating any sound/visuals.

In addition, artists and projects supporting Free and Open-Source Software (FOSS) face another issue: FOSS usually focuses on functionality but is not necessarily accessible to non-tech-savvy users. I.e., high entry fee to use the tool, hard to install/experiment, e.g., LivePose [9]; depends on additional knowledge such as a text-based music programming language, e.g., SATIE [1].

[4] `https://sat.qc.ca/fr/programmation/?c=experiences-immersives`.

[5] `https://sat.qc.ca/fr/nouvelles/sortie-de-residences-metalab`.

3.3 Managing complexity

Hiding complexity is a known principle for information technology and Operational Systems (OSs)[6] . Even though firmware for NIMEs are not as complex as OSs, some operations and procedures may confuse users—in our case, composers and performers. One example of a recurrent operation that is opaque to the user is setting a T-Stick to connect to a specific network. Performers often need to change the network they are connected to according to the venue, and the T-Stick sends OSC messages directly to an IP address in the same network. Changing the connected network can be achieved by reflashing the firmware (i.e., re-programming the device) to replace the network's credentials. This process is tedious, time-consuming, and requires some familiarity with microcontrollers and programming languages. One solution to facilitate these modifications would be to program a web interface for the T-Stick. Users could then connect with this interface using any web browser and change parameters as needed. This implementation can be seen in Figure 4.

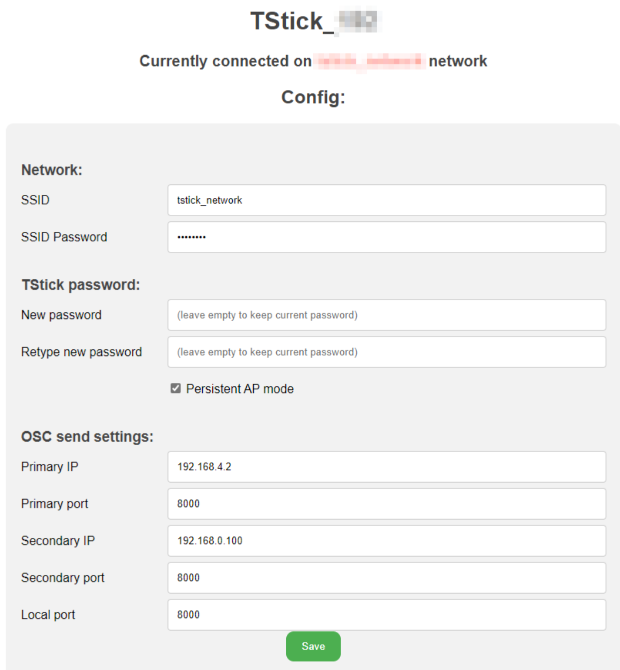

Figure 4: The main screen of the current network manager implemented on the T-Stick. Users can change configuration parameters in real time.

Another interesting example still regarding the T-Stick is how the high-level gestural descriptors used by several performers are now embedded into the instrument and can be accessed directly as OSC messages [10]. The complexity of having multiple Max patches or abstractions generating those descriptors is now hidden from the user. As a side effect, this procedure also makes the descriptor promptly available for users who couldn't run Max patches, e.g., Linux users.

4. SHARING MODULAR COMPONENTS

Not all research topics allow students to work together and increase the reach of each individual's research by doing so. As an inherently interdisciplinary field, music technology

[6] `https://computersciencewiki.org/index.php/Hiding_complexity`.

provides advantages in such interactions. Advancements in instrument design can impact other studies conducted with these devices. Starting in 2016, several researchers from the IDMIL started sharing their instrument design solutions, such as sensor algorithms, user interfaces to configure DMIs/AMIs, or functions to clean and convert data. These solutions were used primarily to accelerate instrument design and enable studies using these instruments.

As part of an investigation on the replicability and reproducibility of DMIs [3], organizing libraries and documentation further advance the process of creating a common set of tools to maintain gestural controllers and creating a plug-and-play method for deploying DMIs/AMIs on stage and research. As of 2021, the T-Stick, GuitarAMI, and Probatio [4] shared embedded sensor fusion algorithms and a large portion of their firmware.

The development of these projects constituted the starting point for creating a library of components, or building blocks, to reproduce devices or share improvements between the DMIs.

5. THE PUARA FRAMEWORK

The Puara framework is derived from modules developed for the three DMIs described in Section 2. The name Puara means "to tie" in Old Tupi[7].

The Puara framework was created as a set of tools for building and deploying new media installations and NIMEs. The first additions included libraries to control network configuration, OSC addresses for the mentioned DMIs/AMIs, and C++ classes to retrieve sensor data in ESP32-based controllers. Using this Wi-Fi manager, entitled Puara Manager, instrument designers could ensure the data would be easily acquired through the network.

Recipes and scripts to quickly configure Raspberry PIs[8] allowed artists and researchers to use gestural controllers as self-contained instruments.

It is interesting to note that even though the authors use the term framework to describe Puara, all libraries, firmware, hardware specifications, and documentation are programmed or created independently and can be used separately. This characteristic allows instrument designers, artists, and researchers to use only the needed elements, connecting with their existing tools and workflows, such as Bela, Satellite CCRMA, or off-the-shelf controllers. In addition, instrument designers can use firmware modules to create their gestural controllers.

The Puara Manager also facilitates instrument usability. Performers can change the controller network and OSC configuration without flashing new firmware into the controller. In addition, high-level gestural descriptors used to interact with the gestural controllers [14] could also be generalized and transferred between DMIs/AMIs. E.g., the jab algorithm from the T-Stick could be transferred to the AMIWrist, and performers could use the same gesture for other instruments, such as augmented drums, as will be described in Section 6.1.

The Media Process Unit (MPU) aims to enable an easier and more reliable performance setup, especially in scenarios where the sound synthesis algorithms and data mapping are configured to run automatically upon initialization, or researchers need software and code ready to deploy or a predefined setup that can be modified or hacked in real-time. A similar scenario is discussed in Secion 6.2. The MPU is generated by a script, compatible with most OSs. The script creates a complete and hackable Raspberry Pi or NVIDIA Jetson OS image that can be deployed for DMI/AMI design by creating self-contained sound synthesis devices, video mapping and immersive environments, or sound installations.

The Puara Framework modules (libraries and tools) are available at `https://github.com/Puara/`, and the first version of Puara's documentation can be found at `https://sat-mtl.gitlab.io/documentation/puara/en/`. Figure 5 shows a short description of the available modules and how they relate to device design (e.g., gestural controller, haptic device) or the MPU.

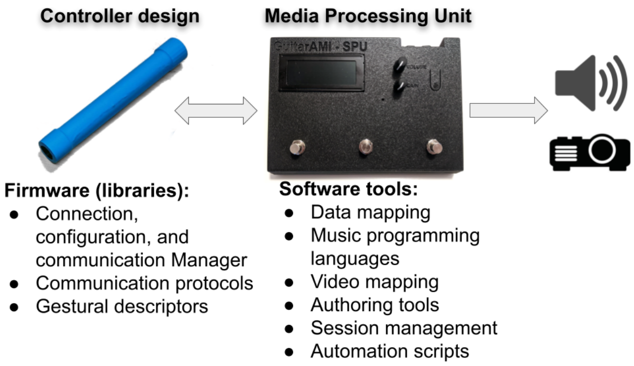

Figure 5: Puara's modules for controller design and facilitate creating self-contained Media Processing Units (MPUs). Modules can be used independently, integrating Puara tools with their existing tools and workflows. A description of the Puara framework can be found in Section 5.

6. USE CASES

Current research and artistic projects using Puara framework components include the already mentioned T-Stick, the GuitarAMI, and Probatio. The list also includes projects such as the t-Tree and the Torquetuner. The t-Tree is a tree-inspired "docking station" for T-Sticks that allows performers to interact with each other in collaborative performances [11]. TorqueTuner is a hardware and software module that allows instrument designers to "map sensors to parameters of haptic effects and dynamically modify rotary force feedback in real time" [12, 19].

The Puara framework was also extensively used in research-creation projects and events involving immersive spaces such as Freeze! and the Kikk festival 2022 workshop as use cases exploring possibilities when using modular frameworks with specific—and diverse—goals.

6.1 Freeze!

The Puara framework was applied artistically in Freeze! (2022), a piece for augmented drum kit—drum kit with added sensors to capture specific performer's gestures—composed by Jason Noble with technical assistance by Edu Meneses and premiered by Martin Daigle at a live@CIRMMT event at McGill University in May 2022, as can be seen in Figure 6. The composer sought to derive all of the sonic material in the piece from live-captured sounds of the kit (snare, toms, kick, cymbals) without external synthesizers or samples. All of the Digital Signal Processing (DSP) began with a spectral freeze, in which a short spectral window is prolonged indefinitely through an interpolated loop. The point in the evolution of the sound's envelope at which the freeze is activated

[7]Old Tupi is an extinct language which was spoken by Brazil's first nations who inhabited coastal regions Southeast of the country. Puara has a more literal meaning of tying (with a rope).

[8]`https://github.com/Puara/MPU`.

Figure 6: Freeze! premiere, May 26th, 2022, at the Multimedia Room (MMR), Montreal, Canada.

dramatically affects the resulting sound quality: as such, the performer needs to be able to accurately control the timing of the freeze. We achieved this control by mapping the gestural data produced by wearable sensors in the shape of wristbands to the freeze trigger.

One goal of this piece was to give the drummer control of parameters not usually associated with the kit, such as melody and harmony. Another goal was to allow the superposition of textures, some electronically sustained and others produced acoustically in real time, so that the performer could manipulate stratified layers (e.g., background, middleground, and foreground) beyond what is normally available from the acoustic drum kit. Finally, the piece aimed to harness the performer's gestures in controlling the signal processing parameters, allowing Daigle to sculpt the sound in real time by moving his hands through the air. These three goals combined effectively made the performer the soloist, ensemble, and conductor of his own mini-concerto.

The AMIWrist was explicitly developed for Freeze! and uses an off-the-shelf open-source IoT development board, the M5StickC. We developed custom firmware for the M5StickC, allowing us to use the Puara library responsible for generating high-level gestural descriptors from raw sensor data. In this particular case, we acquired accelerometer and gyroscope data to generate partial orientation information and instrumental gestures already generalized from the T-Stick, such as jab and shake.

Upon experimentation with Puara's gestural descriptors, we decided to use the jab gesture, created initially for the T-Stick as discussed in Section 5. Other processes such as pitch shift, harmonization, granulation, and bandpass filter manipulation were applied to the frozen spectrum using orientation gestures extracted from the IMU, and discrete gestures using buttons and footswitches. Figure 7 demonstrates how these aspects of the composition were notated in the score. The top "staff" graphically depicts gestures, the middle staff uses normal notation for drum kit playing, and the bottom staff graphically depicts electronic sounds. In this case, the performer jabs horizontally using the right arm to freeze the kick drum, then oscillates that hand to pitch shift the frozen spectrum (counterpointed against a non-shifting spectrum), and finally stops the sound and clears the buffer with a vertical jab (corresponding with another hit on the kick drum). The execution (performance) of is excerpt can be seen at `https://youtu.be/Ib16r6o3uHE?t=35`, while the full dress rehearsal can be accessed at `https://www.youtube.com/watch?v=Ib16r6o3uHE`.

The AMIWrist sends many streams of data corresponding to axes (x, y, and z for raw accelerometer and gyroscope data)

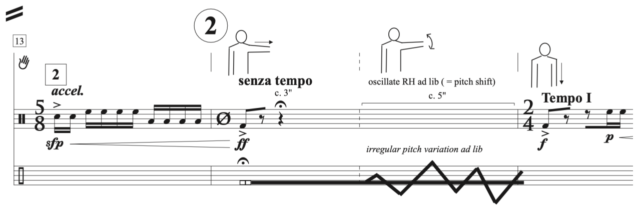

Figure 7: Score excerpt from Freeze! (Noble, 2022) for augmented drum kit using the AMIwrist (Puara wristband).

and other high-level gestural descriptors available through specific Puara framework libraries, e.g., orientation (yaw, pitch, roll), shake, and jab. For Freeze!, the gestural data is sent to a Max patch created by the composer using OSC. The patch also receives audio inputs from microphones capturing various sounds from the kit, so that the wristband data can activate processing in the patch that manipulates the audio before sending it to loudspeakers in real time. The multiplicity of data streams gave the composer a rich palette of gestural information to map onto sound processes, allowing the gestures themselves—including their visual, kinesthetic, and semantic aspects—to become primary compositional elements. For example, the assertiveness or even aggressiveness of a jab gesture stands in stark contrast to the gentle, fluid quality of a slow, free oscillation of the hand. Accordingly, these gestures were used to communicate different affects with plausible cross-modal mappings between gestures, soundworlds, and extramusical domains (e.g., emotions or psychological states).

Freeze! allowed us to experiment with different components of the Puara framework and test the interoperability of each module with tools commonly used by artists. Notably, the OSC protocol allowed the easy connection between the Max patch programmed by the composer and the gestures generated by the AMIWrist. It was also possible to reduce the composer's workload of cooking and extracting meaningful information from raw sensor data. Moreover, the composer could focus on a higher-level description of gestures, allowing easier communication between composer, performer, and music technologist [14].

Another major advantage of conceiving of working with Puara framework components, from the composer's point of view, is that it encouraged a shift from event-based to gesture-based composition. Event-based electroacoustic composition, in which DSP is organized into events that are activated and deactivated throughout the piece, treats the patch more like a musical score, whereas gesture-based composition, in which gestures control DSP without pre-programmed event organization, treats the patch more like a musical instrument.

6.2 Kikk festival 2022 workshop

The workshop presented at the Kikk Festival allowed the participants to explore techniques to create spontaneous immersive spaces, including projection mapping and sound spatialization with a quadraphonic speaker system, using a relatively minimal setup based on embedded computers, as shown in Figure 8. The tools commonly required to create such immersive experiences include software for video mapping, sound spatialization hardware and software, and gestural controls that are either aimed to be controlled by an individual or experienced as a group. During the Kikk festival workshop, we mainly explored two software from SAT/Metalab: Splash for projection mapping running on the

Figure 8: A view of the KIKK festival workshop in Namur, Belgium, in September 2022.

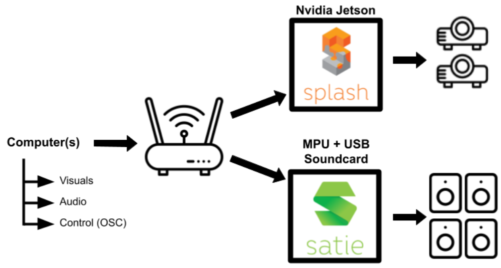

Figure 9: Setup for the workshop.

Nvidia Jetson, and SATIE for sound spatialization running on the MPU.

SATIE is an audio spatialization engine programmed in Supercollider. SATIE is designed to hide some complexity involving the creation and manipulation of audio scenes and sound objects. Splash is a FOSS video mapping software designed to automatically calibrate multiple video projectors and feed them with the input video sources [5].

Developing an immersive space requires substantial technical knowledge and awareness of the environment, as well as creative skills to generate the content to be displayed. Using the MPU in this context helped reduce the technical knowledge required, as it can be seen as a black box that relies on standard communication protocols to be operated. As OSC is now common practice for most creation software, be it proprietary software or FOSS, connecting different tools using this protocol reduced the workload for the workshop participants and organizers alike.

The workshop is part of a series of events organized by SAT/MEtalab aiming at an in situ creation experience of (deployable) immersive spaces. Combining artcraft and tools from multiple practitioners, these workshops invite the participants to actively contribute to the final artistic result, in the form of a small audio-visual or interactive experience.

In this particular event, participants were mostly beginners, with very little technical knowledge of the tools to create immersive spaces. Nevertheless, the organizers relied on already known and mastered software for each participant to minimize hurdles due to the learning curve. Prioritizing the communication between tools also allows for combining all software/expertise capabilities.

Most of the premises presented by the organizers aimed towards using the MPU as a bridge to aid communication between software via the OSC protocol.

Four active speakers were connected to a USB sound card (Focusrite Scarlett 4i4), itself connected to the MPU. The MPU received external sound sources and control data through the network. Sound source transmission was done using JackTrip,[9] while OSC messages carried commands to spatialize different sound sources. The MPU was connected to a router using ethernet cables. Sound sources and OSC commands were coming from a laptop connected with Wi-Fi to the router (called *source laptop* in the remainder of this section). A JackTrip server instance was running on the MPU, and a mono wave file was played on the *source laptop* and sent to the MPU with a client instance. The setup is shown in Figure 9.

The MPU was also running an instance of SATIE that was set up to playback a monophonic stream of sound from the Jacktrip server to the quadraphonic speaker setup, allowing this sound to be positioned in space. With this setup, the most obvious parameter to control was the azimuth position of the monophonic sound source.

[9]https://www.jacktrip.org/.

During the workshop, we presented two examples of controlling a sound object's spatial position using OSC. In each example, OSC messages were sent to SATIE to change the azimuth position of the sound source streaming from the laptop.

- MIDI controller (Korg NanoKontrol): The controller was plugged into the *source laptop*. We used one fader from the controller to set the azimuth parameter. The NanoKontrol sent values between 0 and 127, while the azimuth parameter from SATIE requires values between -180 and 180. To do this value conversion and protocol conversion (from MIDI to OSC), we created a simple script with Chataigne[10]. Running on the *source laptop*.
- Cellphone gyroscope using MultiSense OSC[11]: This free Android application can use the data from multiple sensors of a cellphone and send them as an OSC message over the network. The OSC implementation is incomplete and it requires reformatting to be interpreted by SATIE. This reformatting was done with Chataigne on the *source laptop* and then forwarded to the MPU.

For the Kikk workshop, the organizers could previously prepare the MPU and all entry points for data and audio. The setup process would then be simplified as, once the MPU was set, the ideal behavior is to run all processes automatically upon initialization. At the same time the Puara framework brings hackable tools, the inner workings might be opaque to end users, especially in cases where the devices are prepared beforehand.

7. DISCUSSION

The mentioned use cases provided useful feedback and information regarding the use of Puara, interesting shifts in paradigm suggested by the framework organization, and a list of drawbacks to be addressed in future work. We briefly discuss some key points in the following sections.

7.1 Tools imply paradigms

Regarding the Freeze! project, the composer made insightful observations on how Puara's instrumental gesture library

[10]http://benjamin.kuperberg.fr/chataigne/en

[11]https://play.google.com/store/apps/details?id=edu.polytechnique.multisense.release&gl=US&pli=1

influenced the composition. The composer's previous education and experiences fostered an event-based approach to electroacoustic composition. He had found this laborious, as a composition may involve dozens of events which can each require a great amount of time and work to produce, and also anxiety-inducing, as event-based pieces often involve linear sequencing of events in an order that cannot be easily reversed in live performance, potentially making it difficult or impossible to recover if something goes wrong. On both of these fronts, he found the gesture-based approach implicit in the Puara system to be liberating: if the patch is conceived as an instrument, then the musical structure can flow from exploring different things the instrument can do—as it does in acoustical instrumental composition—without the need to constantly reconfigure the instrument itself.

Additionally, as we can observe in Section 6.1, jab, an instrumental gesture created for the T-Stick, could be repurposed to the AMIWrist. Generalizing instrumental gestures has the potential to not only facilitate instrument design but also assist performers in learning how to play with DMIs/AMIs. The gestures behave consistently across different instruments: The GuitarAMI, the T-Stick, and the AMIWrist have the jab gesture implemented in their firmware. The composer could focus on the performative aspects without creating specific gestural descriptors. The composer was able to choose to focus on adding and modifying the acoustic instrument's behavior rather than creating a static sequence of predefined events.

7.2 The black box effect

A black box is a system that can be viewed in terms of its inputs and outputs, without any knowledge of its inner workings. DMIs/AMIs usually carry some opacity as it is often hard for the audience to infer exactly which sensors are employed for each gesture and if (or how) the sensor data is mapped to generate the perceived output. The MPU potentially brings this opacity to artists and even instrument designers. During the Kikk workshop, participants were unaware of the inner workings of the MPU other than that it contains the software presented during the event.

This black box perception overshadows the function of each tool, to the point that during the Kikk workshop, participants asked about the MPU availability rather than asking for the software they were effectively using to fulfill each task.

The black box effect on the MPU further reinforces the modularity of the Puara framework and the possibility of creating several layers of hidden complexity. One example of this phenomenon can be seen in the several hidden complex layers of Freeze! : The Puara library hid the complexity of creating the jab gesture from the instrument designer, while the designer hid the firmware and gestural mapping complexity from the composer, which hid the complexity of the DSP within the patch. Finally, the performer perceives the augmented drum kit as a single instrument with their instrumental technique and expected sound outcome.

7.3 Feedback and areas for improvement

Compositionally, the dichotomy between event-based and gesture-based thinking and, more specifically, the use of both paradigms in Freeze!, led to some issues. The remnants of event-based organization caused technical problems in the performance that could have been avoided by more fully embracing a gesture-based approach.

Technically, the project team felt the need to explore other hardware options for the gestural controller and processing unit (laptop). It is believed that the MPU could alleviate the need for a computer and reduce setup complexity. However, using the MPU limits access to music programming languages that are not Linux-compatible, such as Max. Additionally, different wristband controllers may have superior functionality, especially concerning battery capacity and sensors to improve orientation estimation.

The high-level gestural descriptors available in the Puara library may be an area for future improvement. In principle, the multiplicity of instrumental gestures from the AMIWrist provides many possible mappings for many different signal processing parameters, which can be very exciting for the composer's imagination. In practice, some of these data are too unreliable or too closely correlated with raw sensor data to be reliably distinguished. For example, the composer found elevation and rotation of the hands to be clearly different gestural concepts and sought to use them to control different parameters of the sound (e.g., elevation mapping onto pitch height, rotation mapping onto harmonicity). In practice, though, the data streams for these two gesture types were not reliably distinguishable, and as a result, a gated footswitch was required to separate them. The input of the instrument designer was necessary to ensure reliable mappings in this instance; ideally, composers would be able to implement their own mappings independently.

At the time of the Kikk workshop, the MPU was in an early prototype stage. Some software was not installed in the created OS image and required some additional work to set, especially without more comprehensive user documentation. For the workshop, JackTrip required manual installation and to be launched using the command line interface. The process of enabling JackTrip allowed the workshop organizers to remove some of the opacity regarding the MPU, showing part of the complexity previously hidden by the device.

8. CONCLUSION AND FUTURE WORK

The Puara framework represents a direct contribution to DMI and AMI design reproducibility research. The libraries and scripts constitute building blocks that can facilitate building new instruments using already implemented code and gestural descriptors.

The framework also allowed enough flexibility for the composer to use the controller and gestural descriptors in the chosen music programming language. This flexibility, however, came at the expense of not being able to use an embedded system for DSP, as stated in Section 7.3. The live performance setup time was also improved compared with a regular performance using electronics.

Future work includes tightening SATIE's integration with the MPU, proving more possibilities for interacting and setting up Puara devices. For instance, Poire[12] could be added as a web interface for SATIE to facilitate the spatializer speakers setup and sound source instantiation. Also, mapping tools, along with session managers for mapping data signals, are a recurring need. Libmapper would be a good candidate for the task, with an interface such as webmapper[13]. Such mapping tools may add more flexibility and avoid the need to install mapping software on each device that requires sending and receiving data.

9. ETHICAL STANDARDS

This work was possible thanks to the SAT, IDMIL, and CIRMMT. This research was also partially supported by a

[12]`https://gitlab.com/sat-mtl/metalab/poire`.
[13]`https://github.com/libmapper/webmapper`.

Discovery grant from the Natural Sciences and Engineering Council of Canada. There are no observed conflicts of interest. All researchers, professional individuals, and workshop participants participated consensually in the activities described in this document.

10. REFERENCES

[1] N. Bouillot, Z. Settel, and M. Seta. SATIE: a live and scalable 3d audio scene rendering environment for large multi-channel loudspeaker configurations. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Copenhagen, Denmark, 2017.

[2] B. Buxton. Artists and the art of the luthier. *ACM SIGGRAPH Computer Graphics*, 101:10–11, 1997.

[3] F. Calegario, J. Tragtenberg, J. Wang, I. Franco, E. A. L. Meneses, and M. M. Wanderley. Open source DMIs: towards a replication certification for online shared projects of digital musical instruments. In *Proc. 22nd International Conference on Human-Computer Interaction (HCII)*, Copenhagen, Denmark, 2020.

[4] F. Calegario, M. M. Wanderley, J. Tragtenberg, J. Wang, J. Sullivan, and E. A. L. Meneses. Probatio 1.0: collaborative development of a toolkit for functional DMI prototypes. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Birmingham, UK, 2020.

[5] E. Décorps, C. Frisson, and E. Durand. Colorimetry evaluation for video mapping rendering. In *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology*, Tsukuba, Japan, 2022.

[6] I. Franco and M. M. Wanderley. Prynth: A framework for self-contained digital music instruments. In *Proc. International Symposium on Computer Music Multidisciplinary Research (CMMR)*, Porto, Portugal, 2017.

[7] S. Freire, G. Santos, A. Armondes, E. A. L. Meneses, and M. M. Wanderley. Evaluation of inertial sensor data by a comparison with optical motion capture data of guitar strumming gestures. *Sensors*, 20(19), 2020.

[8] C. Frisson, M. Bredholt, J. Malloch, and M. M. Wanderley. Maplooper: Live-looping of distributed gesture-to-sound mappings. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Shanghai, China, 2021.

[9] C. Frisson, G. N. Downs, M.-E. Dumas, F. Askari, and E. Durand. Livepose: Democratizing pose detection for multimedia arts and telepresence applications on open edge devices. In *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology*, Tsukuba, Japan, 2022.

[10] T. Fukuda, E. A. L. Meneses, T. West, and M. Wanderley. The T-Stick music creation project: An approach to building a creative community around a DMI. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Shanghai, China, 2021.

[11] L. Kirby, P. Buser, and M. M. Wanderley. Introducing the t-tree: Using multiple t-sticks for performance and installation. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Auckland, New Zealand, 2022.

[12] M. S. Kirkegaard, M. Bredholt, C. Frisson, and M. Wanderley. Torquetuner: A self contained module for designing rotary haptic force feedback for digital musical instruments. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Birmingham, UK, 2020.

[13] C. B. Medeiros and M. M. Wanderley. A comprehensive review of sensors and instrumentation methods in devices for musical expression. *Sensors*, 14(8):13556–13591, 2014.

[14] E. A. L. Meneses. *Iterative design in DMIs and AMIs: expanding and embedding a high-level gesture vocabulary for T-Stick and GuitarAMI*. PhD thesis, McGill University, Montreal, Canada, 2022.

[15] E. A. L. Meneses and M. M. Wanderley. New developments on the augmentation of a classical guitar: Addition of embedded sound synthesis and OSC communication over network. In *Proc. 16th Brazilian Symposium on Computer Music (SBCM)*, São Paulo, Brazil, 2017.

[16] E. A. L. Meneses, J. Wang, S. Freire, and M. Wanderley. A comparison of open-source linux frameworks for an augmented musical instrument implementation. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Porto Alegre, Brazil, 2019.

[17] E. R. Miranda and M. M. Wanderley. *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. A-R Editions, Inc., Middleton, USA, 2006.

[18] A. Nieva, J. Wang, J. Malloch, and M. M. Wanderley. The T-Stick: Maintaining a 12 year-old digital musical instrument. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Blacksburg, USA, 2018.

[19] A.-N. Niyonsenga, C. Frisson, and M. M. Wanderley. TorqueTuner: A Case Study for Sustainable Haptic Development. In *Proc. International Workshop on Haptic and Audio Interaction Design (HAID)*, London, UK, 2022.

[20] P. J. C. Reimer and M. M. Wanderley. Embracing less common evaluation strategies for studying user experience in nime. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Shanghai, China, 2021.

[21] J. Sullivan and M. M. Wanderley. Stability, reliability, compatibility: Reviewing 40 years of DMI design. In *Proc. Sound and Music Computing Conference (SMC)*, Limassol, Cyprus, 2018.

[22] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet. Internet of Musical Things: Vision and Challenges. *IEEE Access*, 6:61994–62017, 2018.

[23] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Computer Music Journal*, 26(3):11–22, 2002.

[24] D. A. Williams, B. Fazenda, V. J. Williamson, and G. Fazekas. Biophysiologically synchronous computer generated music improves performance and reduces perceived effort in trail runners. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Birmingham, UK, 2020.

[25] M. Wozniewski, Z. Settel, and J. R. Cooperstock. A framework for immersive spatial audio performance. In *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, Paris, France, 2006.