Addressing Barriers for Entry and Operation of a Distributed Signal Mapping Framework

Brady Boettcher
IDMIL, CIRMMT
McGill University
Montreal, Canada
brady.boettcher@mail.mcgill.ca

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

Christian Frisson
SAT/Metalab
Montreal, Canada
christian.frisson@gmail.com
cfrisson@sat.qc.ca

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

Joseph Malloch
GEM Lab
Dalhousie University
Halifax, Canada
joseph.malloch@dal.ca

ABSTRACT

The novelty and usefulness of the distributed signal mapping framework *libmapper* has been demonstrated in many projects and publications, yet its technical entry and operation requirements are often too high to be feasible as a mapping option for less-technical users. This paper focuses on completing key development tasks to overcome these barriers including improvements to software distribution and mapping session management. The impact of these changes was evaluated by asking several artists to design an interactive audiovisual installation using libmapper. Observations and feedback from the artists throughout their projects let us assess the impact of the developments on the usability of the framework, suggesting key development principles for related tools created in research contexts.

Author Keywords

libmapper, signals, mapping, installation, feasibility

CCS Concepts

•**Software and its engineering** → **Software creation and management;** •**Human-centered computing** → Interaction design; •**Applied computing** → *Sound and music computing;*

1. INTRODUCTION

The continual creation of new interfaces for musical expression (NIMEs) has resulted in the availability of a large design space of control signals that enable real-time control of audio and visual systems. Each system has its own implementations of protocols and representations for creating mappings to other systems and NIMEs, and often uses common standards such as MIDI and Open Sound Control (OSC) to support compatibility. The limitations of these standards regarding data types, ranges and static signal mappings have led to the creation of mapping support tools, including MetaMallette [6, 9], TapeMovie [16], junXion [17], and O2 [5].

The open-source middleware *libmapper* takes a distributed approach to mapping support, enabling devices to declare their signals independently of one another and without adherence to a representation standard or schema. Discovery, data translation (of e.g., name, type, vector length), arbitrary map processing, and peer-to-peer transport are handled in a distributed fashion over a local network [13]. Users are able to create complex mappings with any of libmapper's user interface tools—such as webmapper (graphical user interface) [19] and umapper (command-line interface)[1]—or by interacting with the libmapper *Graph* API, or by sending session management messages over OSC. While libmapper is written in C, language bindings and environment bridges exist for C++, C#, Java, Python, Max and Pure Data[2], SuperCollider[3], and Ableton Live[4]; with the aim of enabling members of artistic/technological collaborations to continue using tools they are already familiar with rather than asking them to use a common platform.

Over the years, libmapper has been used for both signal mapping research and artistic projects. Research projects include the study of the mapping design process [20], visual representations of mapping networks [19], the extension of map memory to support live looping and sequencing [8], and generalized support for mapping systems and phenomena with multiple instances [14]. Artistic projects using libmapper include the McGill Digital Orchestra [7], Les Gestes [12], and numerous pieces for the T-Stick digital musical instrument (DMI) (e.g., [18]).

Despite this relatively widespread use, most use has been supported directly by the research labs and developers responsible for the libmapper project. Use of the mapping tools "in the wild" is much rarer, partly due to technical obstacles that prevent or slow uptake by composers, instrument designers and media artists who are not supported by a research lab.

This paper attempts to remove the primary barriers for artists and evaluates the changes made by guiding artists in using the framework to design a meaningful artistic project. Recommendations for software developers of related tools

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

[1] https://github.com/libmapper/umapper
[2] https://github.com/libmapper/mapper-max-pd
[3] https://github.com/libmapper/MapperUGen
[4] https://github.com/libmapper/Mapper4Live

are then presented based on the development principles introduced in the paper.

2. ADDRESSING BARRIERS

The libmapper project has shown a clear potential for uses in mapping research in recent years, motivating many researchers to make contributions to the library[5], its mapping interfaces[6,7], language bindings and application bridges[8]. The libmapper library has been embedded into a number of gestural devices as well including TorqueTuner [11], the T-Stick [11] and Probatio [4]. While these contributions bring valuable new features and research attention to the framework, its user base is largely confined to developers with highly technical experience.

We speculate that this characteristic does not stem only from a lack of awareness, but from the technical requirements to install and operate the tools themselves. The fact that the libmapper project aims at supporting users of many languages, programming environments, applications and hardware platforms creates challenges for software distribution, especially since many are "moving targets"—language bindings for Max that were stable and performant in 2012 will not run under newer versions of Max, and web applications (such as the webmapper session manager) are sometimes broken by new web browser releases. The prioritization of new features over stability may also be partly responsible for these consequences, alas, it can prove difficult to justify important engineering work in a research context.

While the framework boasts many benefits when compared to competing mapping approaches, this barrier prevents the user feedback needed to fully determine its feasibility amongst its peers. To open the doors for artistic uses of the framework and to answer research questions regarding mapping design, we shift the focus of development to overcoming obstacles that prevent its common use among artists.

We have chosen to address three main issues that plague the usability of libmapper for the end user. The first concerns improving the ease of installation and setup for libmapper and its bridges across all platforms. Next, we present a mapping session management Python module that handles many of the challenges associated with designing a portable and stable distributed mapping session. Finally, several improvements are introduced to webmapper with the intent of increasing the readability of signals and their metadata for the user. These three developments, detailed in the coming sections, aim to improve the accessibility and usability of libmapper, removing technical barriers to support the pursuit of artistic endeavors.

2.1 Distribution of the mapping framework

The many tools that make up the libmapper framework make its distribution an especially difficult task. We have chosen to focus on supporting continuous integration (CI) of the libmapper library and webmapper, as well as improving the documentation and organization of libmapper's application bridges.

2.1.1 CI and package managers

CI is an important engineering principle that encourages automated building and releasing of software. Often left out of research projects, CI principles reduce debugging time using automated tests and support stability by encouraging frequent releases [10]. Using CI for libmapper and webmapper, we can present users with a precompiled package that can be installed through their chosen package manager and platform. We have utilized Github Actions for these purposes, allowing us to automate the building and testing of libmapper and its language bindings for all platforms when updates are pushed.

Libmapper itself functions as a dynamically-linked (or shared) library, its functions accessed by other programs at runtime. While Linux and MacOS contain dedicated directories in which these types of libraries are installed, Windows leaves it up to the user. For this reason, we did not prioritize the distribution of the libmapper binaries as developers will often want to build shared libraries directly from source. However, we decided to align with Ubuntu's library system and created a personal package archive for libmapper[9] that allows installation using Ubuntu's package manager *apt*.

Now that libmapper's Python bindings are built with CI, we are able to distribute them using pip, Python's package manager. By simply running `pip install libmapper`[10], the bindings for their platform are installed for their Python environment and can be easily updated at any time by downloading the files from Github Actions. As webmapper is built using the Python bindings, users no longer need to build anything from source to operate the application. The creation of standalone webmapper executables for all platforms has been enabled using PyInstaller[11], providing a method of using the application without entering the terminal. A Windows installer has also been created for the standalone version of webmapper, integrating it directly alongside the user's other applications.

Continuing with our design philosophy, we have implemented cross-platform compilation for several of libmapper's application bridges including Ableton Live[12], Max[13] and SuperCollider[14]. For the Max bridge, we have packaged the plugins and submitted them to the Max package manager[15]. If the package is accepted it will become available for one-click installation for all platforms within Max, avoiding all manual compilation or installation. The packages are created automatically with Github Actions, making releases easier for developers as well.

These improvements to libmapper's distribution enable artists to grab the most recent release easily without having to manually build the tools from source in order to use them. Additionally, our release process is much easier and quicker using the virtual build environments provided by Github Actions.

2.1.2 Application bridges

We have implemented a secondary signal creation option for the SuperCollider bridge to provide more flexibility for users. In addition to using signal UGens provided by the

[5] https://github.com/libmapper/libmapper
[6] https://github.com/libmapper/webmapper
[7] https://github.com/bboettcher3/MapperVST
[8] http://libmapper.org/ecosystem/protocol_bridges
[9] https://launchpad.net/~libmapper/+archive/ubuntu/libmapper/
[10] https://pypi.org/project/libmapper/
[11] https://pyinstaller.org/en/stable/
[12] https://github.com/libmapper/Mapper4Live
[13] https://cycling74.com/products/max
[14] https://github.com/libmapper/MapperUGen
[15] https://docs.cycling74.com/max8/vignettes/package_manager

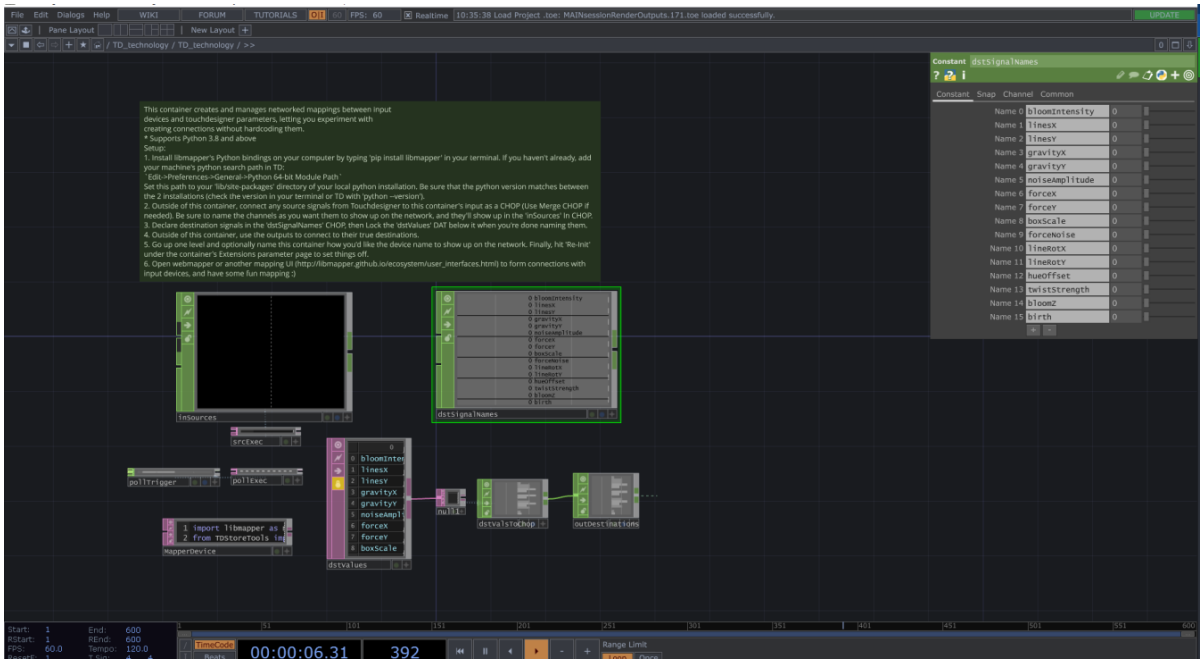

Figure 1: Internals of the Mapper4TD container, showing its inline setup and usage instructions.

plugin, users are now able to route signals using Busses[16], offering the possibility to create signals outside of user-defined synthesizers.

A new libmapper bridge, Mapper4TD[17], has also been recently created for TouchDesigner[18], opening up new opportunities for realtime audiovisual mapping. As seen in Figure 1, Mapper4TD supports both source and destination signal types, letting users send TouchDesigner's many source signals out to other devices as well as modulate TouchDesigner parameters with external devices.

Plugins in TouchDesigner are called *operators*, and are able to connect signals to each other with drag-and-drop wires. TouchDesigner is bundled with its own Python environment, providing abstract structures for developers to use when creating a plugin. Using this environment along with libmapper's Python bindings, we can create a libmapper device and control its signals from within a TouchDesigner operator. One of TouchDesigner's LFO operators was then connected to trigger signal updates at a fixed rate and push them to the network. Finally, Mapper4TD was packaged into a portable container along with detailed setup and usage instructions. As the plugin was designed entirely with Python and TouchDesigner's internal operators, it is cross-platform compatible and requires no compilation.

2.1.3 Documentation and continuity

Another important step we took after these changes were made is to update the libmapper website[19] with the most recent links and documentation for installation and use. We also added a number of recent related projects and publications to showcase the potential of the framework. Usage instructions and build scripts have been updated for many of libmapper's bridges to make them easy to install and use for any platform.

Though the changes presented here may seem obvious, it is important to realize why it has taken so long for them to be implemented to avoid this barrier in related research projects. For one, the rise of CI tools such as Github Actions has only recently opened up the opportunity to automate deployment using virtual environments. Aside from availability, this type of work is not easily justified as research, pushing funding towards feature additions rather than the maintenance and development required to make a project available to a wider audience. We argue that this CI work should be a compulsory part of user-facing research projects to increase the feasibility of the projects for artists.

2.2 Creating a mapping session manager

Webmapper, libmapper's most-used graphical interface for mapping design, contains basic features for loading and saving mapping sessions using JSON structures. To address more complex session management needs and improve the portability of sessions, we have created a mapping session manager Python module that can be used from the command line or imported as a library into other programs.

We call the module *mappersession* and have made it installable through pip similarly to libmapper[20] to align with the CI principles proposed in this paper. Using a Python module permits session management to operate from any Python program or a command line, and work is in progress to extend the module for use in C/C++ programs. Running the session manager from the command line provides a headless mode of session management suited well for automating the setup of artistic works. We have also written detailed installation and usage instructions for the module in its repository[21] and public package to align with documentation standards for Python packages.

2.2.1 mappersession features

The mappersession module maintains backward compatibility with webmapper's JSON structure, allowing the load-

[16] `https://doc.sccode.org/Classes/Bus.html`
[17] `https://github.com/libmapper/Mapper4TD`
[18] `https://derivative.ca/`
[19] `http://libmapper.org`
[20] `https://pypi.org/project/mappersession/`
[21] `https://github.com/libmapper/mappersession`

ing of previously saved mappings with older versions of webmapper. To improve the portability of session files, we have designed a versioned JSON schema for storing information about maps, user interface properties and signal values a user chooses to be initialized once the session is loaded.

Aside from simple loading and saving, mappersession also supports *persistent sessions* to handle devices disconnecting and reconnecting. In a persistent session, the manager monitors the libmapper signal graph and waits until all signals in each managed map are present on the network before loading the map, and reloads the map as necessary as its signals reappear. In a distributed framework like libmapper, this is an important feature for performances with many devices to keep mappings active throughout the whole session.

We have also added the ability to load any number of sessions at once, letting the user change the active session with a libmapper signal representing the session index. By mapping another signal to the index signal, we can explore the effects of cycling through groups of mappings in real time.

2.3 Improving signal readability

Aside from the distribution changes from Section 2.1, we also focused on the readability and usability of the webmapper graphical session manager. Refining these rough edges should make the program (and the framework) more approachable for artists designing mappings. We have also implemented several bug fixes for Windows users, including more readable network interfaces and improvements to network interface selection and clean exiting of the program.

When mapping between signals that do not have their own graphical visualizers, it can sometimes be difficult for users to understand the behavior of the source signals or to tune map processing expressions. For this reason, we have added a simple signal value plotter to webmapper (Figure 2) that can be opened for any signal on the network. A more sophisticated standalone signal plotter that supports multiple signals, vectors and signal instances was also added to the libmapper utilities[22]. This feature has proven useful in the design of devices as well, using visualized sensor values to adjust signal ranges in the firmware of the device.

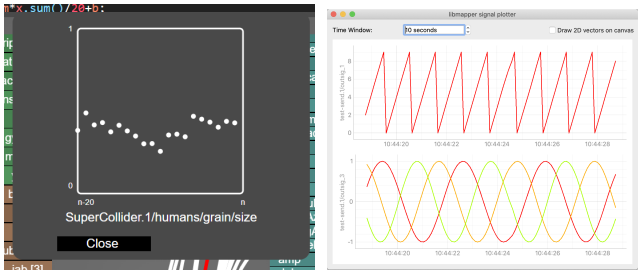

Figure 2: Left: the real time signal value plotter in webmapper; right: a standalone signal plotter.

We also have simplified the signal displays in List and Grid View by moving less-relevant metadata fields to a tooltip rather than showing them in every box. Before this change, each signal displayed its length, range, data type and unit all within its box. While many of these fields are useful when fine-tuning mappings, displaying them all in each signal box can hinder the readability of larger lists of signals. Instead, we have implemented a metadata tooltip that appears when hovering over a signal (Figure 3), providing scalability for new fields and reducing visual clutter.

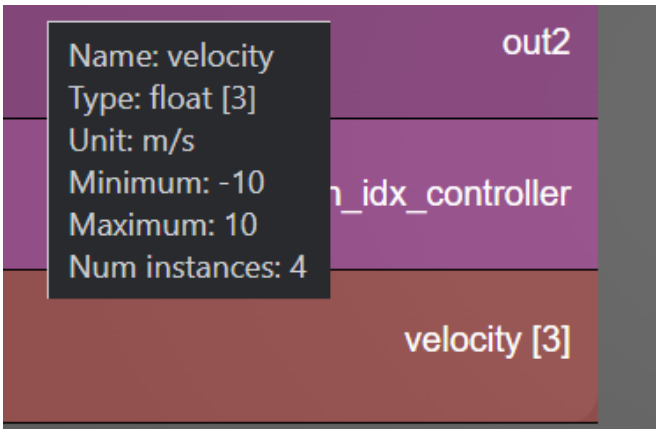

Figure 3: An example of signal metadata displayed as a tooltip

3. EVALUATING THE CHANGES

We intend for the changes reviewed in this paper to increase the usefulness and desirability of libmapper as a mapping option for artists with little to no development experience. To evaluate this claim, we have organized the creation of an interactive audiovisual installation project by commissioning two artists with varying levels of technical experience. The artists were able to make their own design choices to create two distinct interactive sessions over the course of one month but were constrained to using libmapper for their signals and mappings. This project allowed us to evaluate many of the changes to libmapper introduced in this paper including ease of setup and operation, session management and the mapping design process for artists. After describing the installation design process and demonstration, feedback from the artists is assessed to determine the impact of our changes on the usability of the mapping framework and the effectiveness of the development principles imposed.

3.1 Designing an interactive installation piece

3.1.1 Artist profiles

The sound artist is a professional software engineer experienced with SuperCollider for live coding performances and freelance works, using the MacOS platform for their development. The visual artist, with rudimentary Python scripting experience, is proficient with TouchDesigner on their Windows machine and has created interactive patches in the past using the application's integrated LeapMotion and MIDI modules. While neither artist has used a distributed mapping framework like libmapper before, both have experience with designing basic mappings with MIDI and OSC. The diversity in the artists' development experience and platform preference presents a unique opportunity to examine the ease of setup and operation of libmapper throughout the design of the installation.

3.1.2 Devices and programs

The installation was designed using a combination of tools developed at the Société des Arts Technologiques (SAT) and the Input Devices and Music Interaction Laboratory

[22] `http://libmapper.org/ecosystem/utilities`

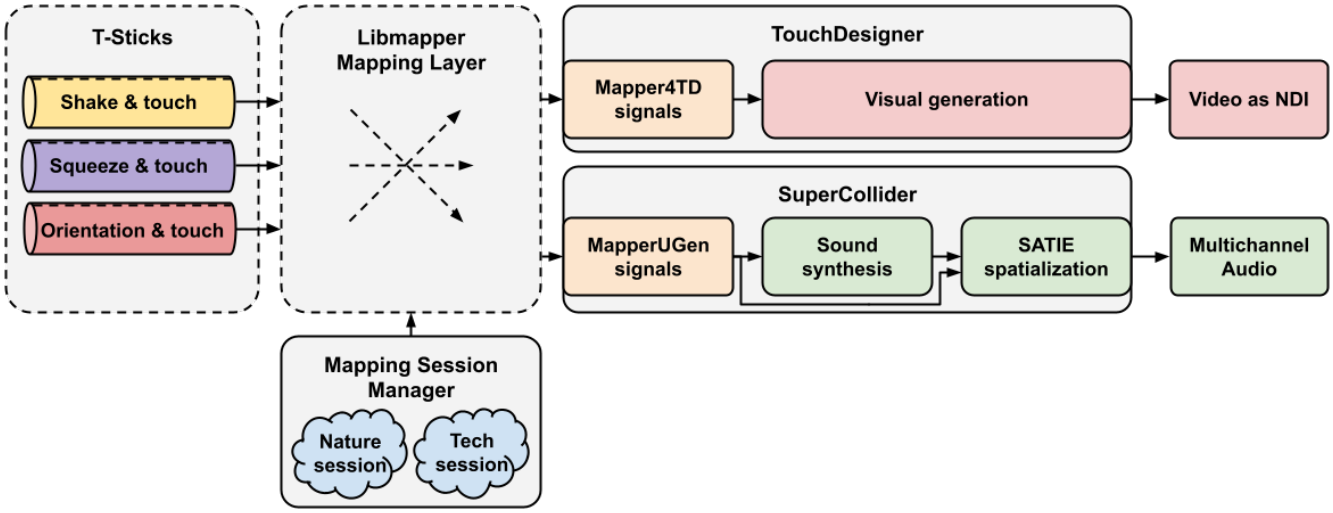

Figure 4: Overview of the systems and interactions used in the evaluation project

(IDMIL). Three T-Stick DMIs [15] were used as input devices, each providing 18 gestural signals made available on the libmapper network and through OSC. The sound artist used SuperCollider for synthesis, adopting the SAT's open-source SATIE plugin [2] for spatialization. The visual artist created projection-mapped visuals with TouchDesigner, enabling us to test the program's new mapping bridge. As each of these systems is compatible with libmapper, we are able to use the T-Sticks to control the audio and visual signals in each program in real time.

3.1.3 The design process

The artists began by choosing a theme to connect the two sessions, deciding upon representing human interactions with nature and technology. The artists then set out to design audio and visual patches for nature and technology, each with their own set of mappable signals. To emphasize experimentation in the mapping design process, the artists separately chose which signals from their programs to expose on the network and left mapping design for when the patches were nearly complete.

Patches for TouchDesigner and SuperCollider were designed by the artists using generic output models for audio and video, utilizing SATIE's abstracted models for sound spatialization. The use of generic output models and distributed mappings makes the project portable to many types of systems with different projection mapping and spatialization requirements.

Mapping design for the two sessions occurred during several experimentation jams throughout the process with the authors and artists, searching for mappings that were meaningful to the piece. The ability of both artists to understand the source signals from the T-Sticks and the destinations in the audiovisual patches was an important requirement for mapping ideation, leading us to create tables with detailed signal descriptions and modulation examples. By virtue of libmapper's distributed approach, mapping ideas could be created and tested easily without the need to change any of the devices or patches.

The artists aimed for the interactions to mirror the theme of each session by using continuous, flowing media and movements for nature and discrete, abrupt ones for technology. The final nature-themed session contained 19 maps and the technology-themed session contained 27 maps, with control divided between three T-Sticks. *Shake*, *squeeze* and *orientation* signals were used with one T-Stick each, keeping the devices consistent in their interaction types. All of the systems and tools used in the installation can be seen in Figure 4. All patches and mappings for each session have been made open source as well to encourage further contributions or reuse on other systems[23].

3.2 Satosphére demo session

As the final stage of the artistic explorations proposed in this project, the artists were invited to set a demonstration of their created immersive space in the Satosphére. The Satosphére, created in 2011, is a full dome with a diameter of 18 meters, and is a permanent immersive modular theatre dedicated to artistic creation and events[24]. This space is capable of 360-degree video projection and a complex speaker setting using 157 speakers grouped to form 31 virtual speakers organized around the dome. These characteristics make the Satosphére an excellent candidate for testing the mapping capabilities of libmapper, webmapper, the bridges developed during this project, and SATIE's ability to adapt to different venues by invoking various spatializer presets.

Moreover, the Satosphére's equipment is configured to provide tools for artists to connect their devices. The connections using audio and video on the Satosphére usually employ Network Device Interface (NDI) as the standard video transmission protocol and MADI-compatible devices for carrying digital audio. This setup allowed the artists to work individually in generating artistic content without concern about how the devices will be connected at the venue.

The same workflow was employed for the data streaming. In that sense, using a decentralized signal mapping framework using libmapper allowed artists to individually work with their content and define which control parameters would be exposed to be subsequently mapped during the demo session. These control parameters were available as libmapper signals, created using the bindings and bridges described in Section 2.1.1.

Simultaneously, the first author could focus on creating mapping sessions based on the libmapper signals (control parameters) provided by the artists. This process allowed modular prototyping while streamlining the setup for the

[23] `https://github.com/IDMIL/Human-Nature-Installation`

[24] `https://sat.qc.ca/en/satosphere`

event. Each artist was able to simply connect with the available network to automatically expose their libmapper signals.

3.2.1 Overcoming on-site technical hurdles

We encountered multiple technical issues during setup. These issues were not caused by the use of libmapper, but resulted from networking and connectivity limitations of some devices. Describing some of the issues we encountered might be useful as a description of commonly found obstacles during interactive demos, performances and installations. The strategies we used to overcome these problems may be relevant and applicable to future projects.

First, we noticed a large number of missing UDP packets when more than one T-Stick was connected to the network. This caused the T-Sticks to intermittently disconnect and reconnect from the network, blocking the mapping session from being fully active. Upon troubleshooting, the issue was attributed to network problems regarding the particular model of microcontroller in the T-Stick. Working with distributed mapping allowed us to program a quick on-site fix in the form of an OSC-to-libmapper forwarding program that discovers OSC signals, creates corresponding libmapper signals on the network, and forwards the signal values through to libmapper. By modifying the point in the network where the T-Stick libmapper signals were created, we could overcome the issue, albeit by adding a bit of latency to the pipeline. Nevertheless, the sessions were finally able to connect all of the mappings on site the day it was developed.

Another hurdle appeared as we realized that the Satosphére requires NDI over Ethernet for video, while libmapper requires Wi-Fi for T-Stick connections. When encountering issues when attempting to use both network interfaces at once, we opted to utilize a Magewell HDMI-to-NDI converter device[25], avoiding the need to use the Ethernet interface on the TouchDesigner machine.

Overall, the setup time and troubleshooting related to mappings were straightforward in comparison with similar events organized at the Satosphére, according to the venue technicians. More importantly, assembling the individual artists' work into a cohesive installation posed little challenge as each libmapper device could be discovered over the network, and the mapping session manager could automatically restore all previously saved connections.

3.3 Feedback from artists

After the conclusion of the installation project, we interviewed the artists to gather information about their experience using libmapper. We explored the effect of technical experience on the ease of setup and use of the framework as well as the resulting feasibility and desire to use it again in the future. Additionally, we employed a well-known usability survey to quantify the usability of the framework and establish a baseline for future work.

3.3.1 Setup and operation

The visual and audio artists claimed that it was simple and quick for them to install libmapper, webmapper and the bridge for their respective programs and platforms. This feat alone speaks to an increase in usability for artists across platforms and technical levels as a result of the distribution and CI principles implemented.

[25] `https://www.magewell.com/products/pro-convert-hdmi-tx`

The audio artist had some initial trouble integrating MapperUGen into their preferred SuperCollider workflow as their method of sequencing continuously created and destroyed the libmapper signals, but we were able to find an alternative using the new signal Bus feature discussed in Section 2.1.2. The visual artist was able to use their basic Python skills to install libmapper and set up Mapper4TD in TouchDesigner without issue by following the instructions in the Mapper4TD container. Each artist went about choosing signals to expose on the network by experimentation, often testing new audiovisual signals with various T-Stick signals to get a sense of the effect on the patch and to find desirable connections for the session.

3.3.2 System Usability Scale results

In addition to the informal interviews, we utilized the System Usability Scale (SUS) survey developed by [3] to establish a quantitative measure of the libmapper framework's usability. The SUS is calculated by aggregating user responses of a short 10-question survey into a score from 0 to 100, and is a useful tool for iteratively assessing the progression of usability in a system with a small sample size [1]. After receiving responses from each artist, the individual SUS scores were 72.5 and 40, producing a mean value of 56.25. One observation from the results is that the artist working as a programmer scored the framework much higher than the non-developer artist. This indicates that the framework requires additional work to become easily usable by artists of all technical levels. As this is the first instance of a SUS evaluation for libmapper, its score may also serve as a baseline for future work toward the same goals.

3.3.3 Desirability of the framework

When asked whether they would be willing to use libmapper in the future, both agreed that it would be a great fit for sufficiently complex artistic projects like this one. Once they had the concept of distributed signals and mappings "figured out", they enjoyed the visual style of designing mappings with webmapper and recognized the benefits of libmapper compared to MIDI and OSC. The positive feedback regarding libmapper's ease of setup and use for different platforms, programs and technical levels points towards an increase in usability from the framework's previous state, and the artists' willingness to continue using the framework speaks to its desirability.

4. CONCLUSIONS AND FUTURE WORK

This paper discussed several obstacles to users when using libmapper, a distributed signal mapping framework, and presented developments that aimed to increase its usability and feasibility for artists without technical experience. From the discussed barriers, we decided to address ease of installation, management of mapping sessions, and readability of libmapper signals. The evaluation project put the changes in the hands of artists with the goal of creating an interactive audiovisual installation.

The developed bridges and language bindings facilitated the use of libmapper by artists inside the tools in which they typically create artistic content (SuperCollider and TouchDesigner) with minimal effort. The developments regarding mapping session management enabled individual artist experimentation and fast connectivity between tools when setting the demo installation. The readability of signals in webmapper aided the final demo setup and allowed the development team to edit and experiment with map-

pings and troubleshoot issues in real time. Updating documentation and utilizing Github's CI tools for automating the building and distribution of the libmapper framework improved its availability and installation process for artists. Feedback from the artists led us to conclude that our developments have strengthened both the usability and feasibility of libmapper for users of all technical levels.

4.1 Future Work

Though these developments are a step in the right direction to support users, there is still more work to be done to improve the distribution and usability of libmapper. Libmapper's bridges should be built using Github Actions for CI and distributed to each community's package manager or repository in order to increase visibility and get more feedback about the framework and grow a user community. Many of the bridges also lack support for creating vectored and instanced signals. These features unlock many new opportunities for mappings, yet have remained largely untested by users due to the lack of support from bridges.

Mappersession, the mapping session manager created during this project, will be expanded upon as well by creating a shared object library similar to libmapper to allow for use in both Python and C programs. Though this will require another large push for distribution and continuous integration, it's important to have a session manager that can be integrated into applications with any language.

The session manager currently makes some unsafe assumptions about the state of the network, which may lead to conflicts when multiple sessions are active. Future versions will let users choose more rigorous methods of map staging instead of persistently reconnecting any disconnected maps, such as one-time staging to initialize a session as devices appear. Webmapper's integration with mappersession will also be developed further to utilize mappersession's ability to save and restore graphical data in sessions. Finally, we are in the process of supporting TCP as well as UDP for communicating map data to allow users to choose between prioritizing reliability or latency for their connections. Additional SUS surveys should be conducted after each development iteration to document the framework's progress toward becoming usable for artists of all technical levels.

5. ACKNOWLEDGMENTS

The authors would like to thank Mitacs and the SAT Metalab for the resources and assistance as well as for providing the opportunity to carry out this research. We would also like to thank Angus MacMinn and Sara Adkins for their hard work and feedback throughout the evaluation project, Lukas and Noah Peterson for their guidance in the development of the TouchDesigner bridge and the Centre for Interdisciplinary Research in Music Media and Technology for its facilities and resources provided.

6. ETHICAL STANDARDS

This work is supported by a Discovery grant from the Natural Sciences and Engineering Council of Canada to the fourth author and a Mitacs Accelerate internship partnered with the Société des Arts Technologiques. There are no observed conflicts of interest. The evaluation project was conducted with two professional individuals who were formally contracted and compensated for artistic work.

7. REFERENCES

[1] A. Bangor, P. T. Kortum, and J. T. Miller. An Empirical Evaluation of the System Usability Scale. *Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[2] N. Bouillot, Z. Settel, and M. Seta. Satie: a Live and Scalable 3d Audio Scene Rendering Environment for Large Multi-Channel Loudspeaker Configurations. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 404–409, 2017.

[3] J. Brooke et al. Sus-a Quick and Dirty Usability Scale. *Usability Evaluation in Industry*, 189(194):4–7, 1996.

[4] F. Calegario, M. Wanderley, J. Tragtenberg, J. Wang, J. Sullivan, E. Meneses, I. Franco, M. Kirkegaard, M. Bredholt, and J. Rohs. Probatio 1.0: Collaborative Development of a Toolkit for Functional DMI Prototypes. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 21–25, 2020.

[5] R. B. Dannenberg and Z. Chi. O2: Rethinking Open Sound Control. page 494, 2016.

[6] S. de Laubier and V. Goudard. Puce muse –la méta-mallette. In *Journées d'Informatique Musicale*, Albi, France, 2008.

[7] S. Ferguson and M. M. Wanderley. The McGill Digital Orchestra: An Interdisciplinary Project on Digital Musical Instruments. *Journal of Interdisciplinary Music Studies*, 4(2), 2010. Number: 2.

[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*, 2021.

[9] E. Ghomi. *Designing Expressive Interaction Techniques for Novices Inspired by Expert Activities: the Case of Musical Practice*. PhD thesis, l'Université Paris-Sud XI, December 2012.

[10] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig. Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects. In *Proceedings of the International Conference on Automated Software Engineering*, pages 426–437. IEEE, 2016.

[11] M. Kirkegaard, M. Bredholt, C. Frisson, and M. Wanderley. TorqueTuner: A Self Contained Module for Designing Rotary Haptic Force Feedback for Digital Musical Instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 273–278, 2020.

[12] J. Malloch. *A Framework and Tools for Mapping of Digital Musical Instruments*. PhD thesis, McGill University, December 2013.

[13] J. Malloch, S. Sinclair, and M. M. Wanderley. A Network-Based Framework for Collaborative Development and Performance of Digital Musical Instruments. In *International Symposium on Computer Music Modeling and Retrieval*, pages 401–425. Springer, 2007.

[14] J. Malloch, S. Sinclair, and M. M. Wanderley. Generalized Multi-Instance Control Mapping for Interactive Media Systems. *IEEE MultiMedia*, 25(1):39–50, 2018.

[15] J. Malloch and M. M. Wanderley. The T-Stick: From Musical Interface to Musical Instrument. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 66–70, 2007.

[16] T. Mays and R. Rubiano. Tapemovie : un environnement logiciel pour la creation temps reel intermedia. In *Journées d'Informatique Musicale*, pages 1–7, Rennes, France, 2010.

[17] STEIM. junXion. Online archive: `https://web.archive.org/web/20201112000658/https://steim.org/junxion/`.

[18] D. A. Stewart and J. Malloch. Everybody to the Power of One, for Soprano T-Stick. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 3093–3096, 2010.

[19] J. Wang, J. Malloch, S. Sinclair, J. Wilansky, A. Krajeski, and M. M. Wanderley. Webmapper: A Tool for Visualizing and Manipulating Mappings in Digital Musical Instruments. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, page 823, 2019.

[20] T. West, B. Caramiaux, and M. Wanderley. Making Mappings: Examining the Design Process. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2020.