Sunflower: an environment for standardized communication of IoMusT

Rômulo Vieira
romulo_vieira96@yahoo.com.br
Arts Lab in Interfaces, Computers, and Everything Else - ALICE
Federal University of São João del-Rei
São João del-Rei, Brazil

Flávio Schiavoni
fls@ufsj.edu.br
Arts Lab in Interfaces, Computers, and Everything Else - ALICE
Federal University of São João del-Rei
São João del-Rei, Brazil

ABSTRACT

The Internet of Musical Things (IoMusT) area, although recent, has well-defined aspects concerning musical practice via the network. However, several challenges are also present, from those related to musical and artistic practice, even those dealing with environmental and social issues. From a computational point of view, the main dilemmas revolve around the lack of resources to deal with heterogeneity and the lack of standard in the communication of the devices that make up this scenario. Therefore, this paper presents Sunflower, a tool inspired by the Pipes-and-Filters architecture that allows communication between different objects, and focuses on its usage protocol. Its layered structure is also presented, showing the types of data, messages, and musical things present in each one of them. After all, the tests and results that certify to the functionality of this environment are demonstrated.

CCS CONCEPTS

• **Networks** → **Network design principles**; • **Computer systems organization** → *Other architectures*; • **Information systems** → Multimedia content creation.

KEYWORDS

Internet of Musical Things, Protocol definition, Pipes-and-Filters, Sunflower environment, Network communication

ACM Reference Format:
Rômulo Vieira and Flávio Schiavoni. 2021. Sunflower: an environment for standardized communication of IoMusT. In *Audio Mostly 2021 (AM '21), September 1–3, 2021, virtual/Trento, Italy*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478384.3478414

1 INTRODUCTION

The Internet of Things (IoT) is a field of study that receives several definitions. Gershenfeld [6] indicates that this area deals with everyday objects with the ability to connect to a data network, capable of handling various devices and communication protocols stacks. Atzori [1], in return, classifies the basic concept of IoT as the generalized presence of objects with common interests, capable of interacting and cooperating. In conclusion, it can be said that the Internet of Things relates the physical and virtual world through network infrastructures and data collections.

Due to its versatility, IoT started to enter several fields of application, such as supply chain management, the health sector, public safety, and more recently, music [8].

When their domains are expanded to the practice of music, the Internet of Musical Things (IoMusT) arises. This area that also uses concepts from human-computer interaction, ubiquitous music, new interfaces for musical expression, artificial intelligence, and participatory art, is a network environment formed by physical or virtual objects to produce or receive musical content. These objects are the so-called Musical Things [18].

The main objective of IoMusT is to assist communication and data exchange between musicians, audience members-musicians, and audience members with other audience members in the context of music creation. It also helps in conducting interactive and/or immersive concerts, remote rehearsals, smart studio production, and music e-learning [5, 10, 18].

Despite being an emerging field, IoMusT already has well-structured visions, but there are also challenges in this field. Among them, we can mention the social and economic ones, resulting from the appearance of new technologies, besides the environmental and the artistic ones [18, 19]. From a computational point of view, one of the main adversities relate to the lack of standardization between protocols, data, and devices present in these environments [18].

Because of this problem, the authors propose a multilayer environment, with an operation mode similar to the Pipes-and-Filters architecture, to help in the communication and interoperability of musical things, defining a protocol responsible for indicating the format of messages, data types, and the behavior of devices present in each layer. The main objective is to offer a new way of structuring these ecosystems to allow gadgets with the most varied features to communicate.

Beyond this initial goal, Sunflower intend to bring a few attributes to IoMusT environments. They are inspired by previous works, be they from IoMusT or network music performance [9, 13, 16, 17].

The first desired characteristic is perhaps the most sensitive point of this work, and we call it *heterogeneity*. This condition indicates that the system must work on multiple devices, with different functionalities and operating systems. At an IoMusT environment, one must also take into account the technical and musical capabilities of the audience members, so that the ecosystem can accommodate users with different skill levels.

This topic is related to *transparency*, which is concerned with directly offering network resources as if they were a local service. Consequently, a lay user might be able to use a remote resource without having to deal with an overwhelmed network configuration. Probably, an interesting *Graphical Display* should be enough to drive the user to this set up. By detailing the information of each musical thing present in the environment, it helps in to connect devices process, as well as in the prevention and correction of failures.

Share Information about the musical things inputs and outputs can also be used to achieve heterogeneity since they can detail technical and musical aspects. When publishing a resource on the network, it is easier to find and connect with those who have some affinity. To a lesser extent, this feature also helps with transparency, since by indicating its properties, it is possible to discover or deduce what the network consumption will be.

At last, integration with legacy software and hardware allows each user to use the tool of their choice, even if it is obsolete or off the production line. For this, there must be conversions between different audio file formats and also the possibility of communication through the MIDI protocol, which is the first industry standard for communication between different instruments.

The remainder of this paper is organized as follows. Section 2 presents the methodology used to obtain latency and jitters, two inseparable aspects of network music practice, as well as a brief explanation of the transmission media that were used. Section 3 conceptualizes how the Pipes-and-Filters architecture works and establishes correlations with musical practice, while Section 4 presents the Sunflower, its layers, and functionalities. Section 5 shows the tests performed and the results obtained. Finally, Section 6 presents a discussion about the protocol and the IoMusT environment, highlighting which characteristics were achieved and the points that could be improved, beyond exhibit summary conclusions about the accomplishments of this work.

2 METHODOLOGY

To assess results on the functioning and behavior of a musical environment that operates on the network, it is important to measure values related to latency and jitter. Latency is the time it takes for information sent from the source host to reach the destination host, added to the machine's processing time, encoding and decoding delays, queuing time, and propagating in both directions. In musical practice, the delay of the sound itself must also be added. One of the main sources that contribute to latency, for obvious reasons, is the distance between the two points that want to communicate. It is also worth highlighting the number of links that intermediate this connection, as well as their transfer and error rates, and package size. Latency could be calculated as the average difference between the relative and the expected time of the note, added to the minimum latency time of the sound [14]. Equation 1 exposes this definition in mathematical terms.

$$latency(\Delta t) = \frac{1}{n}\sum_{i=1}^{n}(t(i) - expected_t(i) + min_t) \qquad (1)$$

Jitter, in like manner, is the measure of delay variation between successive packets. It can also be seen that this delay can cause uneven reception of data. Several factors contribute to this occur, from the physical media for connection (twisted pair, coaxial cable, fiber-optic cable, terrestrial radio spectrum, etc.), to the bandwidth used, and the number of people connected to the network. For better system performance, this value is ideally below 20 milliseconds. If it exceeds 30 ms, the impact becomes noticeable by users, especially in services that use audio and video [3, 4, 7]. Its value is the result of the average latency deviation, as shown in the equation 2.

$$jitter = \frac{1}{n}\sum_{i=1}^{n}|t(i) - \Delta t| \qquad (2)$$

The connection media used in this environment include localhost, twisted-pair cable, and Wi-Fi. The localhost references itself, that is, it forwards data and commands to where the connection was initiated. This type of configuration is useful for testing, where a machine's resources mimic remote resources and the computer acts as a loopback, indicating that processing starts and ends in the same place. Also, the clock for logging the date and time of packages will be the same for both sender and receiver.

A twisted-pair cable is used to connect two computers without the need for interference from routers and switches. It has intertwined wires to cancel electromagnetic interference when transmitting signals. The unshielded model is currently the most used, both in domestic and industrial networks, due to its low price and ease of handling and installation. Care must be taken so that it is not used near equipment that may generate magnetic fields.

Lastly, Wi-Fi predicts wireless communication. One of the advantages of this network structure is the easy connection, as it allows different devices to exchange information with each other quickly and simply. Mobility and stability are also benefits to be highlighted.

3 PIPES-AND-FILTERS ARCHITECTURE

Pipes-and-Filters is a software architecture composed of filters, independent entities capable of processing input data to generate new outputs. They are interconnected by pipes that act as buffers, storing the data and leading them to the filters' outlets. This architecture is widely used in applications that deal with large data flows or in models where little contextual information is needed. It can also work together with other types of architectures [11, 21].

Filters are classified based on their behavior in the system and can act as source, sink, or hybrid. It fits into the first category when it is capable of generating data, such as Filter 1 in Figure 1, and into the second when it only consumes resources from other filters, such as number 4, in the same image. The third and last category covers all those who perform both functions [12].

The pipes, as far as you are concerned, have only the responsibility to transfer data. Occasionally they need to buffer or synchronize activities between filters, managing the amount of information that can be exchanged between them, in order to avoid conflicts, but they will never process or modify the data that pass through them [12].

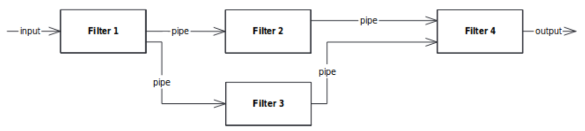


Figure 1: Pipes-and-Filters example [2].

The advantages of thinking about architecture in this way range from fast prototyping, reuse of filters, flexibility, relatively easy implementation, and efficient processing.

3.1 Correlations between Pipes-and-Filters and Music

The operation mode of Pipes-and-Filters in musical context is so vast that it can be observed in several common tools for musical practice [20]. An example is the traditional Moog synthesizers. Created in 1964, they were initially monophonic and consist of separate modules with specific functions, such as oscillating the frequency of a note or applying reverb to it. These functions are analogous to filters, capable of modifying the information that circulates in the environment. Once changed, data flows were sent via cables (which act in the same way as pipes) to the loudspeakers [15].

Other examples are the guitar stomp boxes and pedalboards. The sound generated in the guitar pickups (source filter) are sent through cables (pipes) to the pedals (filters), which apply changes to the signal and send them to the speakers (sink filter), again using cables (pipes).

This behavior can also be observed in audio devices present in studio recordings or at live concerts, where various audio sources, such as microphones and instruments, are connected via pipes on different devices, such as mixers, speakers, and effect pedals. Sometimes a patch bay is used to simplify the addition or removal of new components. These devices can act as a source, sink, or hybrid filters. Like those common to the Pipes-and-Filters architecture, they also don't know their adjacent peers.

Much of the musical creation made in programming languages also follows this working model. Being more explicit in some cases and less in others, Pipes-and-Filters is present in tools such as Pure Data, FAUST, SuperCollider, and ChucK, for example.

4 SUNFLOWER IOMUST ENVIRONMENT

Sunflower is an environment that aims to facilitate the connection of musical things created by different manufacturers, with the most diverse functionalities, data types, and protocols for transmitting musical information. As an environment, it also has operating protocols and a structure that supports all these devices. Such architecture has a way of functioning analogously to Pipes-and-Filters, where musical things behave like filters, without prior knowledge of their neighbors and their respective characteristics. They are also classified as source, when they generate data, sink, when they only consume them, or mixed filter, when they perform both functions. The exchange of information takes place through pipes, which can be both the cables used to connect instruments and the transmission media over the network.

However, thinking about the system in this way generates three main problems: i) great diversity in the data files that will travel through this network; ii) possible occurrences of overloads, and iii) data incompatibility can make filter reuse difficult. To alleviate this problem, Sunflower was divided into layers, a very common approach in Computer Science, used to isolate one layer from the others, making maintenance, alterations, and even the removal of one of them easier. Nevertheless, the environment shown here behaves in a slightly different way, with a parallel operation that separates the layers but allows them to interact with each other when necessary. Each level was categorized according to its musical things, data and protocols, culminating in a digital audio, graphic, control, and management layer.

4.1 Digital Audio Layer

The digital audio layer, as the name implies, is responsible for generating sound data to the environment, supporting audio in Pulse-code Modulation (PCM) format. To ensure the system's interoperability, it must provide information about musical things, such as sample rate, bit depth, number of channels used for communication, ports used in the process, and so on.

This layer also supports a multitude of devices that simulate the behavior of traditional instruments or allow their connection to the environment. These gadgets were created in Pure Data patches and represent the behavior of the audio player, drum machine, loudspeaker, pitchfork, recorder, microphone, and guitar/bass. They send data to the network using the User Datagram Protocol (UDP) and support information in Musical Instrument Digital Interface (MIDI) format.

An important feature of this layer is the ability to divide the processing between its components, adding or removing them in a way that guarantees system scalability and shared music creation. This can lead to improvements in the interactions that take place in artistic performances, in installations, in-studio recordings, in the setup of bands, orchestras, and the like.

4.2 Graphic Layer

Another aspect that contributes to musical performances is the graphic elements, ranging from videos captured in real-time that highlight the behavior of musicians, to animations, figures, special effects, and information that can be transmitted visually, corroborating both for the show's aesthetics and for greater audience participation. For this to occur, the image must go through an encoding and decoding process, so that the visual data can be interpreted by computer systems.

In this context, the graphical layer of Sunflower appears, responsible for this procedure and for ensuring that the system handles this type of data. To this end, the authors again resorted to the multimedia capabilities of Pure Data, using the external Graphics Environment for Multimedia (GEM) to capture and reproduce video data from DVD players, webcams, laptops, or any other compatible medium, besides performing synthesis and real-time image processing.

4.3 Control Layer

Synchronization is an essential factor for devices that exchange data over the network, as well as for musicians, who base their actions on events that take place over some time. Consequently, every musical thing present in this environment must behave in this way, along with the capability to send/receive data and control from the network. This implies a remote control, capable of changing properties such as volume, frequency, beats per minute (BPM), and so on. In Sunflower, this task is performed by the control layer, in particular by the Open Sound Control protocol, thanks to its ability to send UDP packets over the network without the need for prior knowledge of the Internet Protocol (IP), channel, or path. Moreover, it can encapsulate MIDI messages and also send them over the network, contributing to system interoperability.

4.4 Management Layer

In musical presentations, it is common to see the presence of a sound technician to provide support to the musicians. Similarly, computer networks require an administrator, responsible for configuring and maintaining this infrastructure and also providing support to users. In the Sunflower ecosystem, the two roles are inseparable, with the administrator being responsible for ensuring the usability of the network and of the sound and graphic media. For this, it is necessary that musical things can be mapped and that their main features reach the administrator. Nevertheless, this task can be complex, since many of the devices present in these contexts do not even have a graphic interface, processing unit, or some input and output port that provides some information. Additionally, events open to the public are unpredictable in terms of the number of participants, making device mapping even more difficult.

One way to deal with all these issues is through managing objects over the network, where the administrator can see who is connected and what their main characteristics are, to connect only those that can exchange data with each other. For this to occur, each musical thing, when connecting to the network, must inform its ID number, ports used in communication, accepted audio and/or video format, and communication protocols, in order to facilitate the connection. This operation is performed in Sunflower's fourth and last layer, the management layer, represented in Figure 2, which displays the connected devices and their main properties.

The division into layers and its main elements are summarized in Figure 3. It is important to highlight that the Pipes-and-Filters model only inspired the behavior of Sunflower, as this is an approach used for software development and Sunflower is a network communication structure. That said, data exchange takes place through the client-server paradigm, where sink filters play a role similar to that of clients, as they only consume data, and source filters are the servers, responsible for providing the requested information. The advantage of thinking about the system in this way is that a single musical thing can be connected to several other devices, proposing a one-to-many mapping, a counterpoint to the traditional model of connecting objects in musical environments, which offers a wide domain of one-to-one mapping.

4.5 Feasible Scenarios

From the elucidation of how Sunflower works, its layers, data categories, and protocols it supports, some usage scenarios may become possible. The following section, therefore, takes care of showing them, portraying a Jam among a group of people, a studio based on the principles of IoMusT, and an artistic presentation focused on the interaction between musicians and audience members.

Scenario 1 - A jam session with acoustics and musical things

The first scenario is a jam session that combines traditional instruments and electronic devices that exchange information over the network. These instruments can be plugged into speakers or patched, while users/musicians manipulate them from computer systems.

The video can come from the cameras of these same computer systems, such as smartphones and laptops, and sent over the network to SmarTV's or screens, which will show them to all participants. Still, images, videos, animations, and other graphic elements can also be used to ensure an audiovisual experience. The video layer can also be used to control other devices, triggering a sequence of notes on a synthesizer, changing the drum pattern and BPM, and such tasks.

Some users can participate by controlling volume, recording, instrument effects, and turning them on or off over the network. They can also change the color patterns of the images, change their resolution, format, etc.

A sound technician can be responsible for the management layer, handling connections, allowing or disallowing certain users to communicate, as well as choosing which instruments and audio tracks will be sent to the public address.

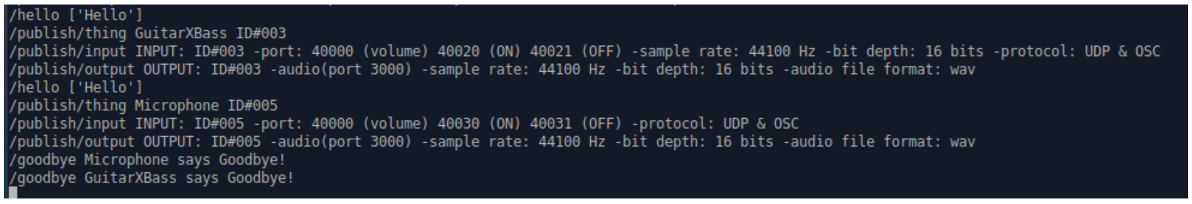
Scenario 2 - An IoMusT based studio

A second possible scenario is a studio that uses Sunflower concepts to record solo artists, duos, and small groups, as well as orchestras with the most varied instruments. For this, the recording interface can adapt its size according to the amount of equipment connected to it. Musicians can play together even if they are not in the same physical location, and audio files can be recorded for later mixing and mastering.

The graphic layer can provide technical information about the network, such as which objects are connected and how to connect to them, and also about the recording, such as displaying sheet music, lyrics, and other information that helps in this process.

Pedal stomps can be reconfigured and controlled remotely or via smartphones. Likewise, the sound engineer can define which channels will be used and combine pre-recorded tracks with live performance. Other musical parameters can be automatically corrected by scripts or artificial intelligence systems. All of this will be done at the control layer.

The management layer will be responsible for connecting the instruments to the recording interfaces, controlling the graphical information that will be displayed, which elements can receive control, etc.



```
/hello ['Hello']
/publish/thing GuitarXBass ID#003
/publish/input INPUT: ID#003 -port: 40000 (volume) 40020 (ON) 40021 (OFF) -sample rate: 44100 Hz -bit depth: 16 bits -protocol: UDP & OSC
/publish/output OUTPUT: ID#003 -audio(port 3000) -sample rate: 44100 Hz -bit depth: 16 bits -audio file format: wav
/hello ['Hello']
/publish/thing Microphone ID#005
/publish/input INPUT: ID#005 -port: 40000 (volume) 40030 (ON) 40031 (OFF) -protocol: UDP & OSC
/publish/output OUTPUT: ID#005 -audio(port 3000) -sample rate: 44100 Hz -bit depth: 16 bits -audio file format: wav
/goodbye Microphone says Goodbye!
/goodbye GuitarXBass says Goodbye!
```

Figure 2: Management screen indicating the properties of a musical thing connected to the environment.

envi.png

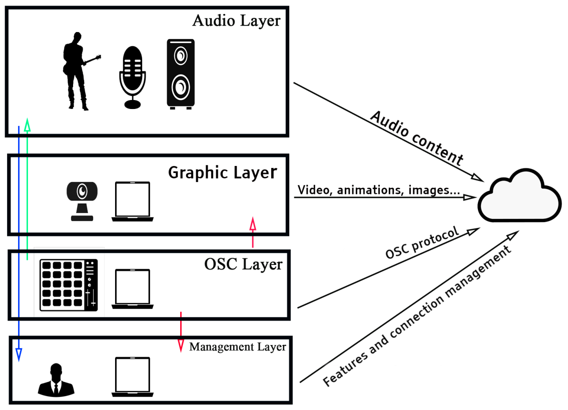


Figure 3: Sunflower’s layered division and its main features.

Scenario 3 - Live performances

Like the two feasible scenarios mentioned, this one that deals with live presentations is also full of possibilities. Musicians can exchange data over the network instead of traditional audio systems, and audience members can collaborate with the musical composition through their own devices.

The graphic layer can be used to convey participation information to the audience, show videos, and animations that expand the artistic capabilities of the performance, and again present some musical aid such as lyrics, setlist, and sheet music to everyone involved.

The control of the guitar pedals, lights, and screens can be done by audience members or by specialized technicians. In the same way, this layer can replace the traditional means of mixing and mastering the sound with digitalized means controlled by the network.

The management layer can restrict access to certain functionality and handle permissions in the environment, whether relating to music or network access. At times, it blends in with the control layer.

5 PRACTICE TESTS AND RESULTS

The tests were performed on an Acer Predator Helios 300 laptop (computer A) for sending and receiving data, and a Dell Inspirion 14 3442 3000 series laptop (computer B), for receiving information only. Computer A used Linux Ubuntu Studio 20.04 operating system, while computer B used a Linux Ubuntu Studio 20.04 LTS. This operating system was chosen for its easy and direct integration with audio systems. In both cases, Pure Data came natively installed. A Behringer C1 condenser microphone, a 6-string electric guitar, and a Behringer U-Phoria UMC202HD audio interface were also used, which allowed the connection of these devices to the computer.

To obtain the values related to latency and jitter in the network, the Wireshark tool was used, capable of analyzing all the traffic and organizing it according to the specificities of each machine involved in the communication. The choice for this software is because it is free, safe, and presents a graphical interface for data analysis.

The first test performed was on the localhost connection. From the programming of the environment and preliminary tests, it was observed that computer A performed better in this aspect. This confirmed the expected fact that the physical configuration of the machine (hardware) directly influences the performance of audio transmission over the network.

Concerning latency and jitter, they were the same as the system itself, regardless of network delay. This delay refers to the time the system takes to pack the audio data, copy it into kernel space, and copy it back into userspace, and unpack. The Pure Data average delay was 1450 μs, but in practical terms, the audio came out almost instantly on the speaker. As a result, the measured values were used as intended values in the calculations performed for the connection by twisted-pair cable and Wi-Fi.

As for the video data, as they use more computational resources, it took a little longer to be displayed on the screen, but nothing that compromised performance. The OSC control data, being small in the number of bytes and being sent in a spaced way, exerted control almost instantaneously, as did the audio. The times recorded for each patch are displayed in the Table 1.

Using the network cable (Table 2), the general latency, that is, the one that includes the values of the audio, video and OSC control layers, was around 1310 μs, while the jitter was 0.4 μs. As for Wi-Fi, it operated on the IEEE 802.11n standard, with a theoretical transmission limit of 150 to 600 Mbps, using an Internet with a bandwidth of up to 100 Mbps and a TP-Link AC 1200 Archer C5 router. The results obtained (Table 3) culminated in a latency of 5400 μs and a jitter of 3711 μs, confirming that the wired connection is better than the wireless one. But even so, the values were within the limits considered ideal, which do not compromise data exchange or musical practice.

6 DISCUSSION AND CONCLUSION

The Internet of Musical Things area brings new perspectives to the musical practice, allowing an interaction between different software and computers in the performance of musical tasks. In this way, a range of possibilities opens up that favor the emergence of new ways of playing, composing, and recording.

Table 1: Results obtained from tests on localhost.

Musical thing	Time
Audio player	0,017 μs
Audio player control	Immediate
Drum Machine	0,017 μs
Drum Machine control	Immediate
GuitarXBassRaw	0,017 μs
GuitarXBassRaw control	Immediate
Loudspeaker	Not applicable
Microphone	0,017 μs
Microphone control	Immediate
Pitchfork	0,017 μs
Pitchfork control	Immediate
Recorder	Immediate
Recorder control	Immediate
Video sender	1 μs
Video sender control	Immediate
Volume	Immediate
Volume control	Immediate

Source: The Author.

Table 2: Results obtained from tests using twisted pair cable.

Musical thing	Expected time	Latency
Audio player	0,017 μs	31 μs
Audio player control	Immediate	9 μs
Drum Machine	0,017 μs	120 μs
Drum Machine control	Immediate	50 μs
GuitarXBassRaw	0,017 μs	600 μs
GuitarXBassRaw control	Immediate	120 μs
Loudspeaker	Not applicable	Not applicable
Microphone	0,017 μs	200 μs
Microphone control	Immediate	120 μs
Pitchfork	0,017 μs	450 μs
Pitchfork control	Immediate	0,03 μs
Recorder	Immediate	Immediate
Recorder control	Immediate	4 μs
Video sender	1 μs	270 μs
Video sender control	Immediate	5 μs
Volume	Immediate	200 μs
Volume control	Immediate	200 μs

Source: The Author.

Table 3: Results obtained from tests using Wi-Fi.

Musical thing	Expected time	Latency
Audio player	0,017 μs	40 μs
Audio player control	Immediate	0,9 μs
Drum Machine	0,017 μs	170 μs
Drum Machine control	Immediate	50 μs
GuitarXBassRaw	0,017 μs	870 μs
GuitarXBassRaw control	Immediate	220 μs
Loudspeaker	Not applicable	Not applicable
Microphone	0,017 μs	740 μs
Microphone control	Immediate	200 μs
Pitchfork	0,017 μs	780 μs
Pitchfork control	Immediate	90 μs
Recorder	Immediate	Immediate
Recorder control	Immediate	0,9 μs
Video sender	0,001 ms	570 μs
Video sender control	Immediate	10 μs
Volume	Immediate	200 μs
Volume control	Immediate	200 μs

Source: The Author.

Despite all its advantages, IoMusT faces some challenges, the main difficulties being the fact of dealing with the heterogeneity and lack of standards of the devices that make up this environment.

Sunflower attacks this problem using an operating protocol based on the Pipes-and-Filters architecture, where elements present in the environment can establish communication with their neighbors without the need to specify their IP address or their audio features, using only the network multicast address to publish their resources. These aspects contributed to the communication of audio and control flows beyond what is trivially used, and these points are the novelties and contributions that this work provides to the area.

From the characteristics initially desired for the environment, all of them were achieved. The 16 patches that make up the Sunflower, combined with its ability to receive the connection of numerous instruments, confirmed the heterogeneity of the system. Likewise, their playful and intuitive use allows different people, with different goals and skill levels, to use them without major problems.

A graphical interface was displayed on the management layer. Despite being simple and relying only on text, it can display the main features of each patch, as well as identify them and report the status of your connection. It displays input and output information, fulfilling another initially intended requirement.

Integration with legacy software and hardware is possible thanks to a patch that encapsulates MIDI information in the OSC protocol and sends it to the network. For this, the tool must be connected to the computer. But once this is done, there is a way to integrate it into the network.

It is worth noting that although Sunflower uses Pure Data as an audio engine and as a support tool for the features present in the other layers, it is not just a set of patches for musical practice in IoMusT contexts, but an environment composed of musical things, protocols, and various types of data. In such a way, these patches only served to simulate devices, leaving the main discussion in the paper in charge of the structure of the Sunflower, in particular its operation model based on the Pipes-and-Filters.

The development of a tool along these lines proved to be a complex job, mainly because it involves knowledge from different areas, such as computer music, the internet of things, computer networks, signal processing, ubiquitous music, and distributed systems. Artistic and social aspects must also be taken into account. The target audience for this tool is formed by musicians, sound engineers, composers, and members of the audience.

Usability, one of the most coveted aspects, was achieved through the publication of resources and the view of the environment as a whole, through the management interface. Latency and jitter, which are inseparable aspects of network musical practice, were within a margin that did not compromise the functioning of the system. However, the mandatory use of Pure Data patches and tests carried out in a controlled environment makes it difficult to indicate an absolute and conclusive result for Sunflower, requiring further tests with a larger audience, when sanitary conditions permit.

Noting the importance of a friendly graphical interface, which can encourage musicians and enthusiasts to experience musical practice using network resources, the authors intend, for future work, to develop a GUI that not only indicates which devices are in the environment but also allows connection between them graphically, instead of doing this only in Pure Data. It is also intended to create a data conversion layer to allow more and more objects with different characteristics to communicate with each other, and also to assign "intelligence" to the devices, allowing them to store information about the objects that they previously communicated with.

Sunflower is still in a development stage, but it has shown promise to meet both musical needs and those related to communication over computer networks. Even so, the authors do not intend to end the discussion of how the IoMusT communication standard should be, in the same way, that they do not claim for themselves the authority to say what should or should not be done when planning an environment along these lines. It is just another contribution that aims to facilitate the expansion of this field, as well as of music and science.

ACKNOWLEDGMENTS

Authors would like to thanks to all ALICE members. The authors would like also to thank the support of the funding agencies CNPq, (Grant Number 151975/2019-1), CAPES (Grant Number 88887.486097.2020-00), FAPEMIG and UFSJ.

REFERENCES

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A Survey. *Computer Networks* (10 2010), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010
[2] A. Bijlsma, B. Heeren, E. Roubtsova, and S. Stuurman. 2011. *Software Architecture.* Free Technology Academy, Washington, DC, USA.
[3] Blog Minha Conexão. 2020. O que é jitter e como ele influencia na sua conexão? https://www.minhaconexao.com.br/blog/jitter/.
[4] Datapath.io. 2016. What is Acceptable Jitter? https://medium.com/@datapath_io/what-is-acceptable-jitter-7e93c1e68f9b.
[5] Angelo Fraietta, Oliver Bown, and Sam Ferguson. 2020. Transparent Communication Within Multiplicities. In *2020 27th Conference of Open Innovations Association (FRUCT).* 61–72. https://doi.org/10.23919/FRUCT49677.2020.9210989
[6] Neil Gershenfeld, Raffi Krikorian, and Danny Cohen. 2004. The Internet of Things. *Scientific American* 291 (11 2004), 76–81. https://doi.org/10.1038/scientificamerican1004-76
[7] Matt Grech. 2018. Acceptable Jitter & Latency for VoIP: Everything You Need to Know. https://getvoip.com/blog/2018/12/20/acceptable-jitter-latency/.
[8] Stephan Haller. 2010. The Things in the Internet of Things. *Bern University* (01 2010).
[9] Joseph Malloch, Stephen Sinclair, and Marcelo M. Wanderley. 2013. Libmapper: (A Library for Connecting Things). In *CHI '13 Extended Abstracts on Human Factors in Computing Systems* (Paris, France) *(CHI EA '13).* Association for Computing Machinery, New York, NY, USA, 3087–3090. https://doi.org/10.1145/2468356.2479617
[10] Benjamin Matuszewski. 2020. A Web-Based Framework for Distributed Music System Research and Creation. *Journal of the Audio Engineering Society* 68 (10 2020). https://doi.org/10.17743/jaes.2020.0015
[11] Regine Meunier, Hans Rohnert, Frank Buschmann, Michael Stal, and Peter Sommerlad. 1996. *Pattern-Oriented Software Architecture, a System of Patterns: 1.* Wiley Press, Hoboken, NJ, USA. 476 pages.
[12] Jorge Ortega-Arjona. 2005. The Pipes and Filters Pattern. A Functional Parallelism Architectural Pattern for Parallel Programming. In *EuroPLoP' 2005, Tenth European Conference on Pattern Languages of Programs.* Proceedings of EuroPLoP' 2005, Tenth European Conference on Pattern Languages of Programs, Irsee, Germany, 637–650.
[13] Flávio Schiavoni, Marcelo Queiroz, and Fernando Iazzetta. 2011. Medusa -A Distributed Sound Environment.
[14] Flávio Luiz Schiavoni, Marcelo Queiroz, and Marcelo Wanderley. 2013. Alternatives In Network Transport Protocols For Aaudio Streaming Applications. In *Proceedings of the International Computer Music Conference.* International Computer Music Association, Perth, Australia, 193–200.
[15] F. Trocco and T. Pinch. 2004. *Analog Days: The Invention and Impact of the Moog Synthesizer.* Harvard University Press, Boston, MA, USA. 368 pages.
[16] Luca Turchet. 2018. Smart Mandolin: Autobiographical Design, Implementation, Use Cases, and Lessons Learned. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion* (Wrexham, United Kingdom) *(AM'18).* Association for Computing Machinery, New York, NY, USA, Article 13, 7 pages. https://doi.org/10.1145/3243274.3243280
[17] Luca Turchet, Francesco Antoniazzi, Fabio Viola, Fausto Giunchiglia, and György Fazekas. 2020. The Internet of Musical Things Ontology. *Journal of Web Semantics* 60 (2020), 100548. https://doi.org/10.1016/j.websem.2020.100548
[18] L. Turchet, C. Fischione, G. Essl, D. Keller, and M. Barthet. 2018. Internet of Musical Things: Vision and Challenges. *IEEE Access* 6 (2018), 61994–62017. https://doi.org/10.1109/ACCESS.2018.2872625
[19] Rômulo Vieira, Mathieu Barthet, and Flávio Schiavoni. 2020. Everyday Use of the Internet of Musical Things: Intersections with Ubiquitous Music. https://doi.org/10.5281/zenodo.4247759
[20] Rômulo Vieira and Flávio Luiz Schiavoni. 2020. In *Proceedings of the Workshop on Ubiquitous Music 2020.* Zenodo, Porto Seguro, BA, Brasil, 109–120. https://doi.org/10.5281/zenodo.4247691
[21] Christian Wulf, N. Ehmke, and W. Hasselbring. 2014. Toward a Generic and Concurrency-Aware Pipes & Filters Framework. In *SoSP.* University of Stuttgart, Faculty of Computer Science, Electrical Engineering, and Information Technology, Stuttgart, Germany, 70–82.