Sunflower: a proposal for standardization on the Internet of Musical Things environments

Rômulo Vieira[1], **Flávio Schiavoni**[2], **Débora C. Muchaluat-Saade**[1]

[1]MidiaCom Lab – Fluminense Federal University (UFF)
Niterói – RJ – Brazil

[2]ALICE – Arts Lab in Interfaces, Computers, and Everything Else
Federal University of São João del-Rei (UFSJ)
São João del-Rei – MG – Brazil

romulo_vieira96@yahoo.com.br, fls@ufsj.edu.br, debora@midiacom.uff.br

Abstract. *The Internet of Musical Things (IoMusT) is an interdisciplinary area of knowledge that aims to improve the relationship between musicians and their peers, as well as between musicians and audience members, favoring concerts, studio productions, and music learning. Although emerging, this field already faces some challenges, especially the lack of standardization and interoperability between its devices. Therefore, the paper presents the design of an IoMusT environment, called Sunflower, highlighting its architecture, protocols, and sound features that can contribute to solving this interoperability problem.*

1. Introduction

The Internet of Things (IoT) is a field of study that consists of the widespread presence of a variety of objects that, from unique addresses, can interact with each other and cooperate with their neighbors to achieve some common goal [Atzori et al. 2010]. When its domains expand to musical practice, the Internet of Musical Things arises.

This area of knowledge is characterized by being multidisciplinary and formally defined as a set of interfaces, protocols, and pieces of information related to music that enable services and applications with an artistic purpose from interactions between humans and musical things or between musical things themselves [Turchet et al. 2018].

A musical thing, for its part, can be delineated as an electronic device capable of acquiring, processing, performing, or exchanging data that serve a musical purpose [Turchet et al. 2018]. The combination of these devices with musical services and applications creates an interoperable environment, responsible for interconnecting musicians, instruments, and audience members, which multiplies the possibilities of interaction in art shows and provides a relationship of interdependence between the participating elements [Vieira et al. 2020].

Despite being an area with great artistic and computational potential, IoMusT lacks standardization in its systems. Although there are some practical implementations of these concepts, they fall into the use of tools that are preferred by their authors, making system interoperability and the adhesion of different people more difficult. This motivated

the elaboration of an ecosystem design called Sunflower[1][2][3], which specifies the structure, protocols, modules, interfaces, and data that meet the requirements demanded by the Internet of Musical Things.

The remainder of the paper is organized as follows. Section 2 presents Sunflower, it is operation mode, protocols, data, and etc., while Section 3 shows the practical implementation of this environment. Section 4 is responsible for displaying and discussing the results achieved. Finally, summarized conclusions about the accomplishment of this work are exposed in Section 5.

2. The Sunflower environment

Such an environment was conceived to suggest a new way of thinking about IoMusT systems and to indicate possible solutions to the most recurrent problems in this area, such as the difficulty of dealing with heterogeneous devices and not being comprehensive for people with different technological and musical knowledge. Sunflower can support numerous instruments and musical things, along with exchanging information over the network, which provides an experience that can be tested and inferred from results.

As for their desirable features, they can be divided between those that deal with **the environment** and those that concern **the devices**. For the first category, the authors used definitions from the IoMusT area and the concepts that permeate it [Turchet et al. 2018]. Consequently, a good network structure for artistic-musical presentations must present some general aspects, such as **low latency**, **interoperability**, and **scalability**. Further, there are concepts such as easy user integration, assimilation of different types of data, and independent implementation of the tools. Due to the difficulty of meeting these requirements in a real system, Sunflower focuses on meeting the first three topics mentioned.

As for the desirable features for the devices, aesthetic, expressive, and ergonomic factors are relevant, since it is a field with artistic concerns. In addition to heterogeneity, it is also important that the information be made available in a graphical interface.

2.1. Thinking About Pipes-and-Filters

Sunflower has an operation mode analogous to the Pipes-and-Filters architecture. This is because such architecture allows reuse, replacement, and evolution of the system. In this way, each musical thing will have a behavior correlative to a filter, being independent entities that receive data in their inputs, process them and send them to their neighbors, whose priorities are not previously known. The ducts are responsible for only transmitting data, not applying any type of processing or modification to them. By bringing this model to the Internet of Musical Things, it is possible to create a separation of roles and suppose the flow of information in the environment. With this, it is possible to define that entities that only generate data are classified as **source**, while those that only consume are named as **sink**. When they play both roles, they are classified as **hybrid**. It is important to note that despite being inspired by Pipes and Filters, network implementation still takes place through the client-server model.

[1]Manuals and documentation are available at: `https://github.com/romulovieira-me/sunflower_iomust_environment`

[2]How to Setup Sunflower: `https://youtu.be/lGW5eobrYwc`

[3]Sunflower Live Action: `https://www.youtube.com/watch?v=bWfVCGAda_c`

This way of thinking about the environment ensures that it is heterogeneous since different devices can be present and communicate with each other, besides ensuring ease of inclusion and/or removal of objects. However, this form of structure is faced with three problems arising from the diversity of data that travel through it: i) the type of the data must be agreed between the modules; ii) possible overloads may occur when trying to standardize data; and iii) the incompatibility in the data type can make it difficult to reuse filters. To solve this problem while maintaining the main features and contributions of pipes and filters, Sunflower was divided into 4 distinct and independent buses, presented in detail as follows.

2.2. Thinking About Buses

Several tools in the area of computer science use a bus architecture, like hardware components (USB), network protocols and inter software communication. Generally, this architecture is used to isolate implementation details from one device to another, providing some maintainability for the system. In such a way, the buses in Sunflower are also independent, being characterized according to the musical things, data types and protocols present in them. These buses are:

- **Digital Audio Bus:** As the name implies, it is responsible for exchange digital audio data in the environment. An important feature of this bus is the ability to divide the processing between its components, being able to add or remove them in a way that guarantees the scalability of the system and the musical creation in a shared way, resulting in improvements in the interactions that occur in these environments;
- **Graphic Bus:** Another aspect that contributes to musical performances is the graphic elements. This bus, therefore, is responsible for ensuring that the system is able to handle visual representations;
- **Control Bus:** Responsible for providing to the environment remotely control methods, capable of changing properties such as volume, frequency, beats per minute (BPM), etc. These features are modified by musical things designed exclusively to play this role;
- **Management Bus:** Allows the system administrator to visualize who is connected to it and what its main features are, only connecting those that can exchange data.

2.3. Protocol Proposal

The devices connected to the above mentioned buses are responsible for providing services to the network. However, these services only indicate the operations that a device in certain bus is able to perform on behalf of its users, but do not inform how these actions are implemented. Therefore, it is important to have a protocol that represents a set of rules that controls the format and meaning of the messages that circulate through the system.

Setup Protocol

In Sunflower, each entity at each bus uses a configuration protocol to implement its services. This regulation is simple and must be followed by the components, where its main foundations are: allow any node to indicate its status to the network manager; always synchronized this status with the network; make each node display its main features and ports capable of both receiving and sending data; connect only a pair or set of nodes

Credentials	*/hello/number_ID/address_IP/human_name*
Input information	*/input/ID_number/port_type/musical_thing_type/sample_rate/bit_depth/ audio_file_formatnumber_of_channels/sender_IP/port_number/ port_description/network_protocol/_musical/information_protocol/ human_name*
Output information	*/output/ID_number/port_number/port_type/port_description_receiver/ port_number/receiver_name*
Network setup	*/network/ID_number/device_name/IP_address/current_port _number/new_port_number/current_multicast_address /new_multicast_address*
Audio setup	*/setup_audio/ID_number/device_name/IP_address/bit_rate /bit_depth/file_type/number_of_channels*
Graphic setup	*/setup_video/ID_number/device_name/IP_address/codec/display _resolution/color_pattern/file_type*
Creating Logical Devices	*/create/ID_number/device_name/IP_number/ new_musical_thing/audio_features/video_features*
Deleting Logical Devices	*/delete/ID_number/device_name/IP_number/ new_musical_thing/audio_features/video_features*

Table 1. Step 1 and 2 - Discovery and Devices setup messages

that actually support this connection and report that it is being connected or disconnected from the network. From these definitions, the Sunflower setup protocol was divided into 5 steps, presented and conceptualized below.

Step 1 - Discovery In this first step, Sunflower will behave like a traditional discovery protocol, looking for other nodes on the network and establishing contact between them. For this, each musical thing, when connecting to the network, must inform, through an Open Sound Control (OSC)[4] message, its unique identification number, IP address and human name, according to the pattern shown in the sequence. A credential message needs to be delivered to all devices in the environment, which should also answer with their credentials. This eliminates the need for an administrator to go through any data exchanges in the system. Next, the input ports, responsible for receiving data, must be reported. As for the output ports, they must reach nodes capable of receiving this data. For this, they use the pattern seen below. It is worth highlighting that each element differs from the other because of its features, but these messages are comprehensive enough to inform the entire system of the main characteristics of each device present in it, creating the necessary parameters to some communication takes place. The messages to be sent are presented in Table 1.

Step 2 - Devices setup: The second stage of the protocol allows eventual changes in the network settings and/or device data. In this way, all attributes informed in the resource publication can/must contain features messages, which will be answered with environment update announcements, indicating the current status of each element. Also taken part of this step 2 are the messages to create logical devices, which are nothing more than a replication of a musical thing present in the system. In that wise, after a given artifact

[4]Protocol used in communication between computers, sound synthesizers and other multimedia devices.

Connecting ports	*/connect/ID_number/device_name/IP_address/input_number/output_number*
Performance Mode	*/setup/ID_number/device_name/IP_address/actual_status_/new_status*
Configuration Mode	*/performance/ID_address/device_name/IP_address/actual_status/new_status*
Goodbye message	*/goodbye/ID_number/IP_address/device_name/status*

Table 2. Step 3, 4 and 5 Messages

receives this message, it generates a modifiable copy of itself, allowing the audio, video (when possible), and network features to be modified to satisfy the users' needs. After confirming the duplication of an element, the system should answer with a state update, similar to the one sent when the settings of musical things are changed. The message that creates this copy can be seen below. If users want to delete a logical device, it must send a message in the example seen below. In response, you will have again a status update. These messages can adopt patterns presented in Table 1.

Step 3 - Setting up and connecting the environment: With the devices set up, it is time to configure and connect the environment. The messages must, therefore, inform their credentials, the port through which the data will be sent and the input port of the device that receives this information, according to the pattern presented in Table 2.

Step 4 - Mode Change: Musical things can follow another method of operation when being used in practice, called **performance mode**. In this mode, the devices no longer accept remote setup, which guarantees greater technical and artistic stability to users, while also facilitating network management. This entails the need to switch from configuration mode to performance mode and vice versa. In this context comes the fourth and penultimate step in the Sunflower protocol, which deals precisely with this change in the behavior of musical things. When the first message is sent, it changes the whole environment and not a single musical thing. As a consequence, the devices receive an answer that indicates that they are now "unmodifiable", that is, no longer manageable by users. Table 2 presents the messages responsible for this.

Step 5 - Close the connection: Finally, the last step of the protocol permanently disconnects the musical thing from the network, where it will no longer be able to exchange data. Table 2 presents this farewell message.

Performance Protocol

This protocol results from the change in the operating mode in the device configuration and is responsible for dealing with the types of data exchanged in the Sunflower buses. This results in several implementation possibilities and tools for each of these levels, and the authors do not intend to limit this behavior.

3. Implementation

Considering the structure and protocols that govern the operation of Sunflower, its four buses were conceived in a practical way. The system has 27 prototypes of musical things. They were developed in the Pure Data (version 0.50.2), chosen for being multiplatform, open-access, easy to program, and for being present in several systems that deal with music over the network or in real time.

As for sending data to the network, it happens through the UDP and OSC protocols. This was possible thanks to the use of an external[5] called **mrpeach**, capable of encapsulating packets with these types of data and sending them to the network in a simple and fast way.

To further expand the artistic and interactive possibilities of this system, four digital art were created in the Processing language[6] to be connected to the graphic bus. It allows changes in colors, speeds, and directions movements of the digital art thanks to the **oscP5** library, which receives OSC messages responsible for these changes, providing new ways of interacting and controlling the environment.

In the control bus, every musical thing can be managed by the network. To fulfill and manage these and other requirements, a Command Line Interface (CLI) was created in the Python language (version 3.8.10). From a functional point of view, the CLI presents information about the IP address and ID number of each musical thing, as well as its input and output data. It consists of four commands: -i, capable of individually listing the ports and input data of musical things; -o, responsible for displaying the ports and parameters of the output data; -c, in charge of connecting the IP:port pair, in a way that eliminates the need to do this on the device itself; and finally, the -d command, which finishes the connections.

The CLI runs directly in a terminal. As it also acts as an OSC server to receive messages from musical things, if no command is entered, it displays only a list of the device type (with an identification number, IP address, and human-readable name), a second list with its input settings, and a third list with information about the outputs of musical things. It is worth mentioning that the CLI must be run before devices connect to it. Otherwise, even if they are correctly connected to the network, their information will not be displayed, which can cause problems in the management and functionality of the system.

Finally, three musical things (drum machine, microphone and tuning fork) were transferred to smartphones, to be played in the MobMuPlat[7], in its version 1.84. In this way, users gain a new platform to interact with the environment, while Sunflower accepts an easy and general-use device, allowing the integration of more people.

With all of that exposed, Figure 1 shows an idealized environment for Sunflower. Note that it is composed of several elements, such as a guitar, effects pedal, audio systems, smartphones, and wearables. Each one will play a role exposed in Section 2.1, that is, they can be source, sink, or hybrid. Once connected to the network, they tell the manager what types of data and protocols they can receive and send through their ports. From this knowledge, the administrator can interconnect those who have something in common and who will be able to cooperate with each other. Sunflower works by proposing how discovery and setup messages should be, classifying musical things according to how they

[5]Additional function for the Pure Data developed by the community, with operation mode similar to libraries in traditional languages.

[6]Created by Ben Fry and Casey Reas in 2001, it is a modification and simplification of the Java language, removing the original aspects that require a deeper knowledge about programming. In this way, it becomes light and easy to learn, ideal for visual and digital art.

[7]Standalone app for iOS and Android capable of hosting and running Pure Data code. It can also perform sound synthesis, receive MIDI and OSC data via the network, display images and much more.

work, and dividing them into buses. All this increases the heterogeneity of the system and allows the inclusion of diverse artists and participants.

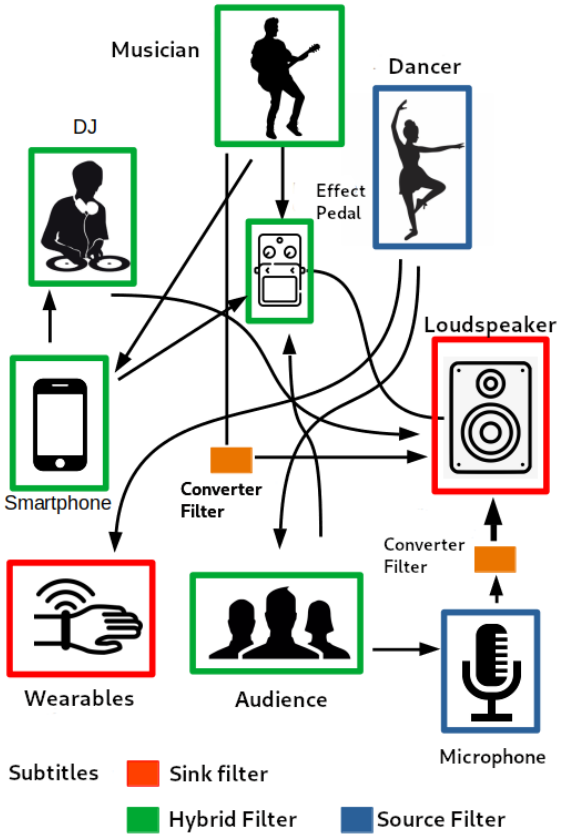

Figure 1. Environment idealized following the design concepts of Sunflower.

4. Discussion and Results

About initially desired features for the Sunflower, all were achieved in implementation. The system proved to be heterogeneous when supporting 27 musical things (simulated on Pure Data), including devices capable of generating and consuming graphic information, in addition to presenting the circulation of a wide variety of data.

The command line interface, despite being simple and relying only on textual information, manages to display the main characteristics of each musical thing, as well as identify them and report their respective connection states, making them show to the system all their netowrk features. It displays information about inputs and outputs, fulfilling another initially intended requirement.

The operation similar to the Pipes-and-Filters architecture was also evident since the musical things had no prior knowledge of the other elements connected to the network, being only responsible for processing the data and sending it to the output. The fulfillment of all these requirements opens the way for the system to present a playful and intuitive use, allowing multiple users, with different goals and skill levels, to use them without major problems.

Finally, it is noteworthy that although the proposed protocol is present in the application layer of the TCP/IP model, it can also be implemented at other levels, making

the system similar to a mesh network. This changes perspectives on a recurrent issue in real systems, which is bandwidth usage. Even so, the idea of functioning analogously to the Pipes-and-Filters architecture is maintained, predicting the existence of source, sinks, and hybrid filters, ensuring that the five steps of the protocol are met.

5. Conclusions

The Internet of Musical Things area provides new perspectives for musical practice, mediating the interaction of musicians with their peers or of musicians with audience members through different computational resources. Live concerts, studio recordings, and music learning tend to benefit from this new trend. Despite the positive points, some issues are evident, such as the difficulty in dealing with device and data heterogeneity. The Sunflower comes up precisely to tackle this issue. By proposing an operation mode analogous to the Pipes-and-Filters architecture and messages for discovery and setup, it is possible that each musical thing present in the environment can establish communication with its neighbors without the need for prior knowledge of the IP address or audio/video data and protocols they use. In addition, they do not impose communication restrictions due to their physical characteristics.

The development of this platform proved to be a complex activity, as it requires knowledge in several areas of computer science, such as computer networks, signal processing, software engineering and sound design, as well as concepts from the music area. This paper aimed to contemplate the artistic-musical creation process and its target audience is made up of sound engineers, technicians, musicians, music teachers and students, scientists who are interested in research in the area and audience members.

The tool proved to be promising in meeting the musical and network needs, but even so, it is not intended to end the discussion about which data and protocols should be used in IoMusT communication, in the same way, the author does not claim for himself the authority to say what should or should not be done when planning an environment along these lines. Such a tool should not be treated as something monolithic, but as an open solution, capable of receiving contributions from other users and developers, as well as being incorporated into your own projects.

Acknowledgments

Authors would like to thanks to all MidiaCom Lab and ALICE members. The authors would like also to thank the support of the funding agencies CNPq, (Grant Number 151975/2019-1), CAPES (Grant Number 88887.668167/2022-00), FAPEMIG and UFSJ.

References

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer Networks*, pages 2787–2805.

Turchet, L., Fischione, C., Essl, G., Keller, D., and Barthet, M. (2018). Internet of musical things: Vision and challenges. *IEEE Access*, 6:61994–62017.

Vieira, R., Gonçalves, L., and Schiavoni, F. (2020). The things of the internet of musical things: defining the difficulties to standardize the behavior of these devices. In *2020 X Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 1–7.