Rebuilding and Reinterpreting a Digital Musical Instrument - The Sponge

Ajin Tom*
IDMIL,CIRMMT
McGill University
ajin.tom@mail.mcgill.ca

Harish Venkatesan*
IDMIL,CIRMMT
McGill University
harish.venkatesan@mail.mcgill.ca

Ivan Franco
IDMIL,CIRMMT
McGill University
ivan.franco@mail.mcgill.ca

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

ABSTRACT

Although several Digital Musical Instruments (DMIs) have been presented at NIME, very few of them remain accessible to the community. Rebuilding a DMI is often a necessary step to allow for performance with NIMEs. Rebuilding a DMI exactly similar to its original, however, might not be possible due to technology obsolescence, lack of documentation or other reasons. It might then be interesting to re-interpret a DMI and build an instrument inspired by the original one, creating novel performance opportunities. This paper presents the challenges and approaches involved in rebuilding and re-interpreting an existing DMI, *The Sponge* by Martin Marier. The rebuilt versions make use of newer/improved technology and customized design aspects like addition of vibrotactile feedback and implementation of different mapping strategies. It also discusses the implications of embedding sound synthesis within the DMI, by using the Prynth framework and further presents a comparison between this approach and the more traditional ground-up approach. As a result of the evaluation and comparison of the two rebuilt DMIs, we present a third version which combines the benefits and discuss performance issues with these devices.

Author Keywords

digital musical instruments, gestural controller, embedded sound synthesis, self-contained DMI, mapping

CCS Concepts

•**Hardware** → **Sensor devices and platforms;** •**Human-centered computing** → **Gestural input;** •**Applied computing** → *Performing arts;*

1. INTRODUCTION

Availability and longevity are key factors that determine the success of an instrument [16]. However, for digital musical instruments (DMIs) [14], there are various challenges to overcome, for example, unpredictable advancements in technology, problems related to hardware and software compliance, lack of documentation [13], etc. which could lead to obsolescence or abandonment of the instrument. Deprecated or aging electronic components, communication protocols (RS-232, USB 2.0, Bluetooth, etc.), operating systems updates, upgrades, and software compatibility are all potential causes of DMI malfunction.

Ferguson and Wanderley [4] have stated that one effective measure for the evaluation of a DMI is its ability to reproduce a performance of a particular piece. For the preservation of several works and re-interpretation of pieces, the existence of the DMI in a working condition is necessary. Though there might be multiple versions of certain DMIs, for instance, the Hands [17] or the T-stick [10], if one wishes to play these instruments, they would probably need to rebuild the instrument. Though rebuilding a DMI might be a hard task, mostly for non-experienced developers, it also provides an opportunity for customization in order to satisfy one's idiosyncratic needs.

In this paper, we discuss various aspects involved in the development of our interpretations of *The Sponge* [11] by Martin Marier, a DMI embedded with sensors to detect squeeze, flexion and torsion along with buttons to form an interface to generate and sculpt sounds. The key idea of the sponge is to harness the properties of a retractable, flexible object that gives the performer wide range of multi-parametric controls [8] with high resolution in a maximized gesture space, considering its high maneuverability.

2. OVERVIEW

We created two versions reflecting our interpretation of the original instrument:

- **Version 1**: Similar in concept to the original instrument, but implemented with different mapping strategies for sound synthesis and augmented with vibrotactile feedback using Libmapper [9] and the Vibropixels [6]. Version 1 uses an open architecture, where the user can choose from a diverse set of tools with no constraints in terms of the choice of hardware, software, or protocols.
- **Version 2**: A re-interpreted table-top version of the sponge with embedded sound synthesis, implemented using Prynth (v0.3) [5], a framework that allows for easy development of DMIs with up to 80 analog sen-

*Equal contribution

NIME'19, June 3-6, 2019, Federal University of Rio Grande do Sul, Porto Alegre, Brazil.

sors (such as pots, switches, FSRs, LDRs, etc.) as inputs and synthesizers built on SuperCollider, responsible for sound synthesis and mapping.

One of the main differences between Martin's Sponge and the two versions introduced in this paper is the spatial multiplexing of different synthesizers and virtual instruments, where the controls of each set of sounds exist in different regions of space, i.e. the orientation of the sponge determines the mapping between the sensors and musical parameters. This feature can possibly be used to achieve some level of transparency as compared to using buttons for mode switches in multi-modal instruments [3], which make mode selection arbitrary and hidden from audiences.

In the following sections we discuss the rebuilding process and modifications made to the design aspects of the two versions. Section 3 describes the design and implementation of version 1 along with addition of vibrotactile feedback to the DMI. Section 4 presents the implementation of version 2 on the Prynth framework. Finally, section 5 presents comparisons and evaluation of the two approaches and introduces a third version that combines the advantages of both.

3. VERSION 1 : REBUILDING PROCESS

In this section, we present the key design aspects and modifications implemented while building the first version.

3.1 Hardware design

The Sponge's supporting structure is made of flexible foam, so the first step was to choose the optimal material. The important considerations kept in mind were to make sure that the material chosen had a reasonable amount of retractability so that gestures like twisting and flexion could be carried out easily. The dimensions (24cm x 16cm x 4cm) were chosen such that the embedded force-sensing resistors (Figure 1) had enough foam material above the sensing area (depth) and the orientation sensors on either side of the foam were placed reasonably far (along the breadth). Several cuts had to be made in the various layers to allow strategic placement of the various sensors.

The advent of superior sensing technologies today provided us the opportunity to make use of IMUs (Inertial measurement units - comprised of accelerometer, gyroscope and a magnetometer) instead of simple accelerometers that were used in the original design. However, the main motivation behind this upgrade was linked to our artistic goal of being able to produce multi-timbral polyphonic sounds out of the various gestures using dynamic mapping strategies. In this way, the performer could make quick tilts and sways to re-connect control signals to different synthesizers and mappings. Two IMUs were placed at opposite ends of the block of foam at the sites of the performer's hands in order to detect flexion and torsion from the roll and pitch angles of the IMUs.

On the other end of the foam, a pair of FSRs separated by a distance of 7cm, connected end-to-end by a rigid strip, perform functions similar to that of a slide bar. Position of press on this strip was measured by the relative pressures detected by the FSRs (Figure 2).

Another design improvement involved doubling the number of tactile switches and placing them on the bottom face of the sponge in the form of 4 x 2 matrix right under an FSR placed in the middle layer. The main aim here was to mimic the feel of playing on a touch-responsive instrument; here the buttons would trigger notes and the amount of force applied on the FSR would be mapped to velocity control. Some switch buttons are also configured to trigger

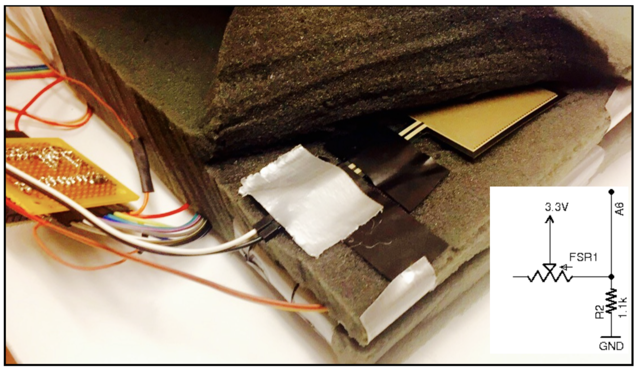

Figure 1: Single FSR placed within the first layer

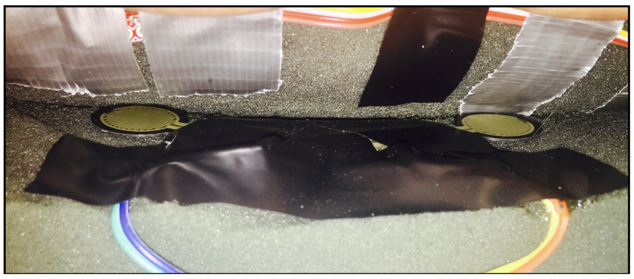

Figure 2: Two small FSRs placed at a distance above which a slide bar is placed

sequences, control a looper and change presets and mappings. (Figure 3)

A Teensy 3.2 microcontroller performs signal acquisition by collecting data from all the sensors connected to the GPIO pins and I2C buses. All the sensor data is transmitted via an HC-05 Bluetooth transceiver and received on the laptop's native Bluetooth port. All intermediate calculations and signal conditioning, such as implementation of filters, sensor fusion computations of roll, pitch and yaw from the IMU data and position of press on the slide bar are carried out on the Teensy before transmission, hence reducing the amount of data to be transmitted. A vibrotactile feedback actuator is placed in the front face of the foam(explained in Section 2.3). Care had to be taken for the number and positions of vibrotactile actuators attached to the body of the foam, since the vibrations would cause undesirable fluctuations on the IMU data. Block diagram of the arrangement is shown in (Figure 4).

3.2 Mapping and Sound Synthesis

As mentioned in earlier sections, our main artistic goal is to perform polyphonic electro-acoustic music. Considering the maneuverability of this DMI, it was really useful to have quick mode/patch changes with gestures such as rotating the sponge along the planar axis. The magnetometer keeps track of number of rotations made; one full circle of rotation to the right switches the patch number up by one and left to switch down. This way the performer would be able to have different sound synthesis mappings with the ability to change them quickly to generate polyphonic sounds. Considering the different kinds of sensors that could measure force applied, elevation, tilt amount and switches triggered, the performer had several degrees of freedom while allowing both separable(mode switches) as well as integral(control tremolo depth while changing LFO parameters) control. An example from one of our performances include: triggering a huge percussion hit (jab: z-axis acceleration) with a long re-

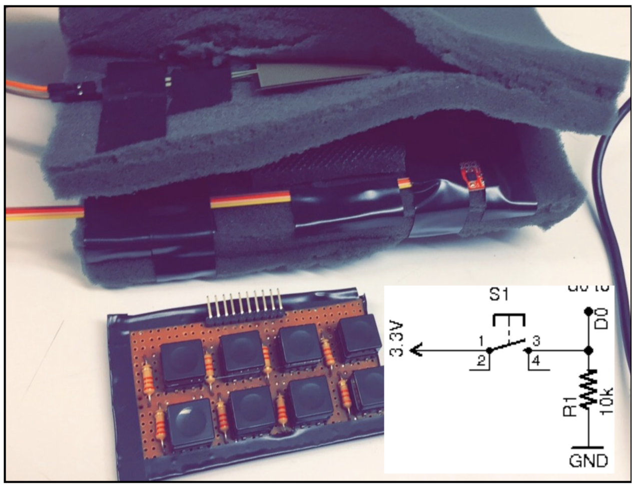

Figure 3: Placement of IMU (above) and switch matrix (below)

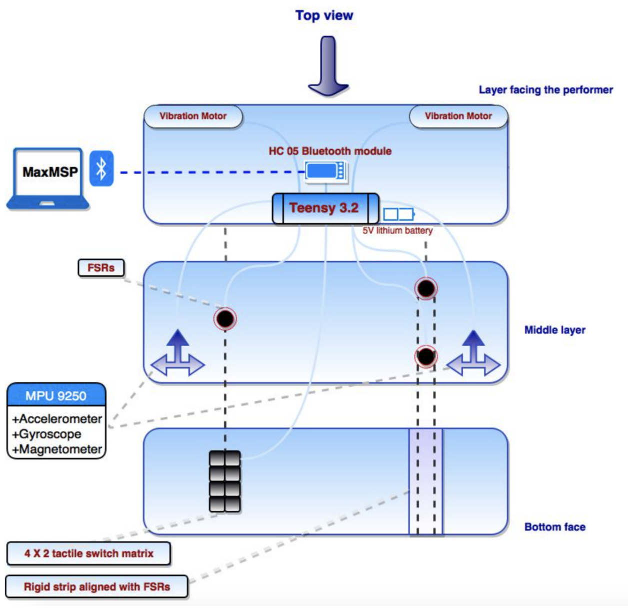

Figure 4: Block diagram: Version 1

verb (adjusting the dry-wet mix amount using the FSR) and quickly rotating the instrument once to play a melody while triggering a stereo-delayed drone sound using the switch matrix. This kind of spatial multiplexing of various synthesizers and sound samples allow dynamic [15] mappings that could change over time.

A Max/MSP patch running on the host computer receives all sensor data serially via Bluetooth. The Max patch manipulates the sensor data to normalize the magnetometer values and also performs signal conditioning and scaling of sensor data so that the output values are compatible in the 8-bit MIDI CC (0-127) range. This Max patch communicates to the VibroPixels Max patch via Libmapper to send control signals for vibration feedback and sound synthesis.

Libmapper [9] is a cross-platform software library for declaring data signals on a shared network and enabling arbitrary connections to be made between them. Libmapper creates a distributed mapping system/network, and potential for tight collaboration, easy parallelization and interactive control of media synthesis. Here Libmapper is used to switch between various mapping strategies easily. The mappings have 2 parallels: one from the driver Max patch to the Sound Synthesis patch and other to the respective vibration feedback patch depending on the feedback mechanism chosen.

3.3 Vibrotactile feedback

It is generally accepted as a fact that performers of traditional musical instruments receive important feedback from their instruments through the sense of touch [2]. In the context of DMIs, we implemented three different feedback mappings using the VibroPixel [6] (Figure 5) which enhances the 'feel' of the instrument. The Vibropixel also includes RGB lighting, which can be used as an additional visual feedback mechanism to improve the awareness of various modes of the DMI and it also adds to the aesthetics of the DMI. Three different kind of feedback mapping were implemented:

- Sign-based mapping provides discrete cues to help the performer navigate the DMI in multi-dimensional space. This becomes very useful when the DMI has to be brought to absolute positions and orientations.
- Signal based mapping creates a haptic illusion to get a better 'feel' of the instrument [12] by producing vibrations using perceptual sound features in the audio output produced by the DMI.
- Cue-based mapping uses tactons [1] for improvisational suggestion [7] during sponge-duet performances.

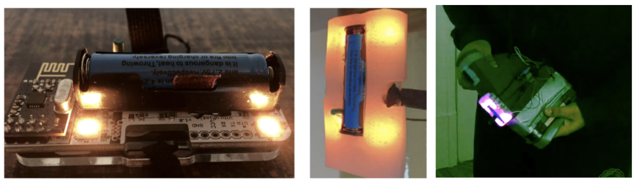

Figure 5: The Vibropixels

3.4 Challenges faced in version 1

Apart from having to retain the flexibility and maneuverability of the sponge (placement of non-flexible sensors and circuitry), the technical challenges we faced in our approach to developing version 1 are:

- Developing the firmware (which included setting up serial communication, sensor data acquisition, sensor fusion and filtering algorithms, etc.) required reasonably high level of expertise in hardware design aspects and associated programming skills.
- In an open-architecture set up like in version 1, there were no well-defined set of software that could communicate with each other. There were several issues relating to non-compatibility due to version updates and lack of robust protocols to setup communication between Max/MSP, Libmapper and the sound synthesis patch.
- There were quite a few instances during performances when technical issues popped up and they were difficult to trace, considering there were several separate components in the system.
- This DMI lacked quick start-up and immediate playability. Bluetooth pairing, loading of patches and setting up four different applications ended up being time-consuming as well as processor-heavy on the computer.

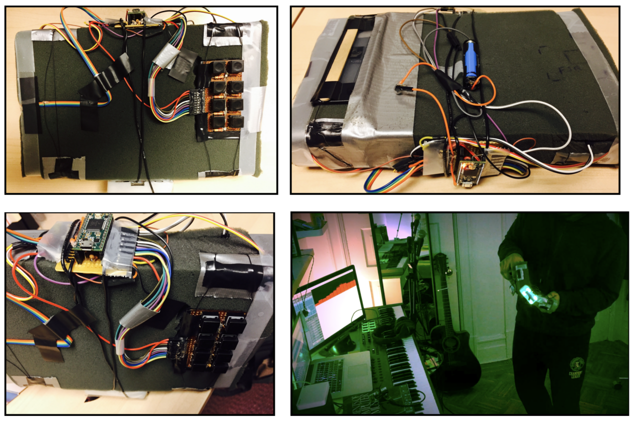

Figure 6: Final prototype design of FlexSynth in action

4. VERSION 2

The roots of the challenges mentioned in the previous section can be traced majorly to the fact that there were many sub-systems, forcing the developer to build links between them. Some amount of abstraction in development of a DMI can help alleviate this problem and assist in faster development of prototypes. Our hunt for a good foundation to develop a DMI led us to Prynth. The Prynth framework [5] provides a platform for easy prototyping and development of musical instruments by employing simple plug-and-play methodology for interfacing commonly available analog sensors with a sound synthesis engine in a portable form. The documentation provided gives sufficient guidance on how to go about building instruments from scratch using the framework.

The Prynth framework helped in overcoming a lot of hassle faced in building version 1. Firstly, since the system is optimized for capturing gestures and synthesizing sounds, the setup time is far less compared to version 1. Secondly, the main goal of Prynth is to integrate the sound synthesis engine of a DMI within the interface to form a self-contained instrument, much like hardware synthesizers whose qualities stimulate attention towards the cognitive activity of performance. An added advantage of using Prynth is that it makes use of SuperCollider on a mobile platform for sound synthesis.

4.1 Hardware Design

The Prynth module consists of a Raspberry Pi 3b, which carries out several signal processing tasks such as sound synthesis and mapping of sensor data to sound synthesis parameters, with a Teensy 3.2 microcontroller for the sensor data acquisition. Since the Prynth base hardware is larger than the Teensy 3.2 used in version 1, it cannot be used in a highly maneuverable context such as the original sponge. This encouraged us to reinterpret the instrument by utilizing the core principle of the sponge, which is to use gestures such as flexion, torsion and squeeze to control sound synthesis, in a table-top instrument with a larger form-factor (foam of dimensions 30cm x 15cm x 10cm).

This version comprises of the same sensors as used in version 1. Though the increased size of the DMI restricts its maneuverability, an additional FSR was included in the available real-estate (middle of the sponge), thereby increasing its gesture space. Several sensor placement strategies were considered, one such design is shown in (Figure 8). The sensor placement strategy employed in this version of The Sponge was such that the instrument would be used

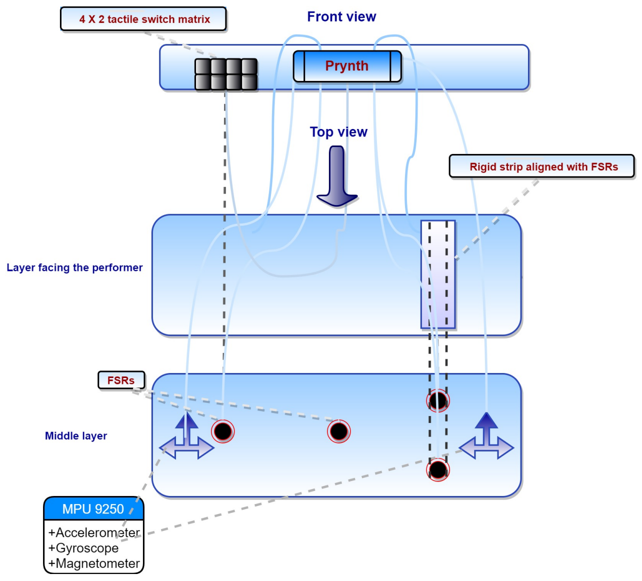

Figure 7: Block diagram : Version 2

like a surface controller, which would be placed on top of a table while performing. (Figure 7) shows the block diagram of version 2 and (Figure 10) shows the playing position of the instrument.

The Arduino code provided with Prynth (v0.3) offers a basic structure for analog sensor acquisition along with simple low-pass filtering of the sensor data [1]. The user would only have to enter the number of multiplexers being used and number of sensors connected to each multiplexer in specified places in the program and the sensor data can be accessed within SuperCollider. Since the IMUs used in this project communicate through I2C protocol, the code provided could not be directly used. The program was modified to include initialization of and data acquisition from the two IMUs. Moreover, in order to reduce the amount of data flow from the microcontroller to the Raspberry Pi, the roll, pitch and yaw information was computed within the microcontroller itself.

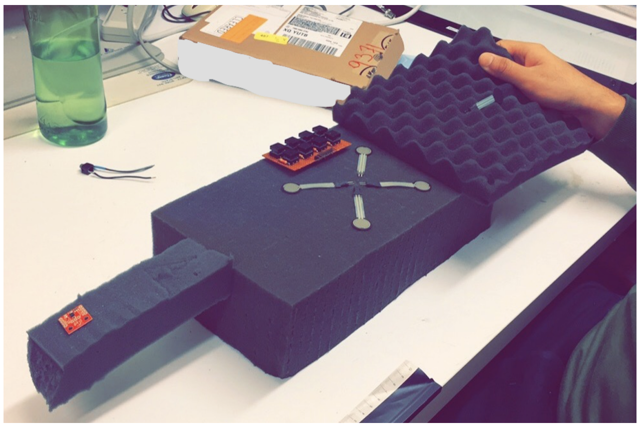

Figure 8: Alternate designs considered

4.2 Sound Synthesis and Mapping

The sound synthesis part of the instrument was programmed on SuperCollider through the web interface of Prynth. For

[1]Current version of Prynth, v0.5, was not available at the time of development

demonstration of spatial multiplexing, three different synthesis types were chosen; FM synthesizer, Granular synthesizer and Sampler.

The difference between the implementation of spatial multiplexing of synthesizers in versions 1 and 2 is that in version 1, synthesizer control changes occur on complete rotation of the instrument, whereas, in version 2, because of less mobility (confined to the surface of a table), the rotation of the instrument to switch between synthesizers was restricted to 180 degrees. (Figure 9) shows the state transition diagram for mode changes.

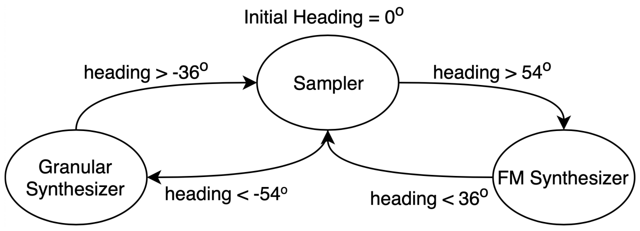

Figure 9: State transition diagram for mode switches

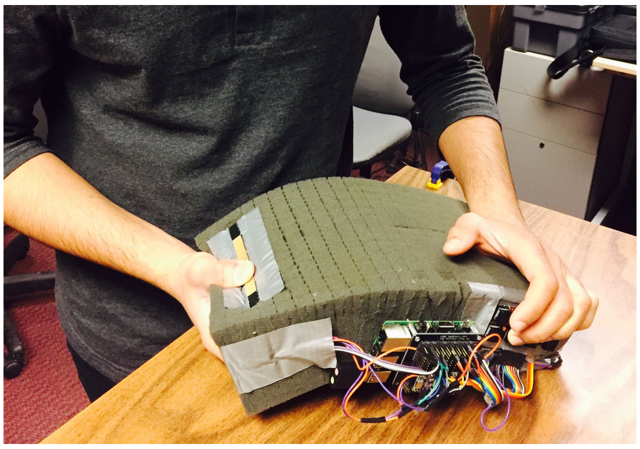

Figure 10: Playing position : Version 2

5. DESIGN APPROACH COMPARISON AND THE SPONGE V3

In this section we present an informal comparison of the two versions of The Sponge, version 1 built from ground up and versions 2 built using the Prynth framework, on two aspects; difficulty in development and ease of use. From a development standpoint,

- Version 1 has less power requirements (350mAh Li-ion battery to power the Teensy and Bluetooth module for more than 12 hours), hence is a wireless controller without the sound synthesis part of the DMI. On the other hand, the Prynth framework consists of the full DMI and uses a Raspberry Pi which requires more power, hence it would require a wired powering setup.

- The Prynth framework takes care of most of the hardware aspects, hence designing version 2 did not require significant experience in electronic hardware design. Whereas, version 1 needed a fair amount of expertise in embedded system design as it involved interfacing sensors and implementing filtering algorithms.

- Form factor of the Prynth module is not suitable to comply with the original design of The Sponge, while version 1 has potential for miniaturization to match the original design. Version 1 is more maneuverable, thus allowing more gestures that use spatial multiplexing.

From an usability standpoint,

- Version 2 is a self-contained DMI and it requires little to no set up time to get it running. One would only have to plug in the power supply and wait for the system to boot to performance state. On the other hand, the setup for version 1 involves setting up bluetooth communication, loading mappings on libmapper, setting up DAW with virtual instruments and routing MIDI and audio buses, which is time consuming.

- Version 2 favours the use of SuperCollider for sound synthesis and mapping schemes, whereas on version 1, one can simply modify the MIDI signal routing and mapping schemes to generate sound using any synthesis and mapping tools like Max/MSP, SuperCollider and even VSTi plugins running within a DAW.

- In version 1, feedback is required as mode switches happen due to the absolute orientation of the device. Whereas, since version 2 is confined to a surface, feedback is not as critical.

In this context, a third version was built which combined the desirable features of versions 1 and 2 which are:

- The mobility and manoeuvrability of version 1.

- The ease of development and cohesive nature of subsystems of version 2.

Version 3 was built using a modified version of Prynth, where the sensor data acquisition part (Teensy microcontroller) was physically decoupled from the sound synthesis unit (Raspberry Pi). The sensors were connected to the Teensy directly using a perfboard instead of using the PCB that comes as part of the Prynth package in order to satisfy the size constraints for maneuverability. Figure 11 shows the sensor data acquisition unit of version 3.

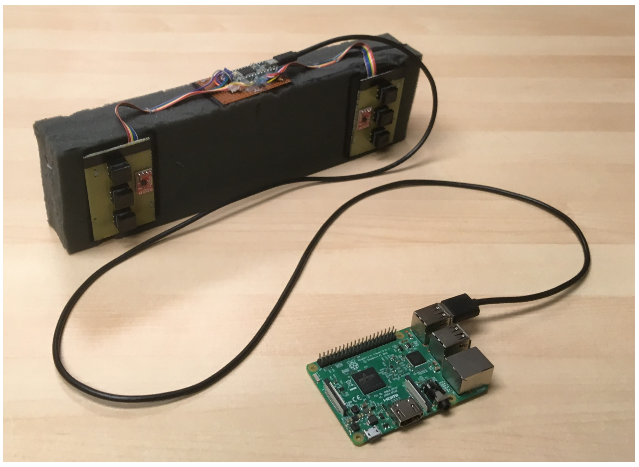

Figure 11: The Sponge: version 3.

This unit is directly connected to the Raspberry Pi via USB, although this could be accomplished using an HC-05 connected to the teensy transmitting sensor data to the onboard Bluetooth receiver of the Raspberry Pi, which would result in increased mobility and the similar playability as that of version 1, but introduce latency due to the wireless transmission.

This version consists of 1 IMU, 1 FSR and 3 push buttons on either edges of the instrument as shown in figure 11. This was done deliberately to explore how the number of sensors would affect the ease of learning and playing the instrument. We will not be commenting on these aspects as they are out of the scope of this paper. The sound synthesis is however similar to that of version 2.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the various aspects that are involved in the process of rebuilding and reinterpreting a pre-existing DMI, The Sponge, along with design changes and modifications to re-purpose the instrument to the needs of the performer. These modifications implemented helped us achieve spatial multiplexing of synthesizers and provide vibrotactile feedback to the user in case of version 1.

We successfully rebuilt the instrument using two approaches; version 1 made use of an open architecture in which there are no restrictions on the choice of sensors and their interfacing schemes, and version 2 made use of the Prynth framework, which provides a guided pathway to build DMIs by handling most of the hardware design aspects. We also presented a comparison of the two approaches, evaluating the two instruments in juxtaposition, which helped in designing a third version that incorporates the best aspects of both. Using a framework reduces the threshold of entry by making the process of building an instrument easier. Prynth, being an open-source framework, provided an excellent starting point to design an instrument and at the same time gave us enough freedom to make modifications to suit our needs.

Through the course of development of this project, we realized that rebuilding a DMI is important for two reasons. Firstly, it helps us gain a better understanding of an existing DMI and identify aspects that can be improved using newer technology, apart from facilitating innovation. Secondly, the development of an instrument takes place through multiple iterations of rebuilding and reinterpreting it. Hence, rebuilding is a necessary step for the evolution of DMIs, which would effectively tackle issues concerning obsolescence and longevity.

Our future work would be focused towards development of a miniaturized version of the Prynth framework that would be more suitable for instruments with size constraints and greater mobility. Such a framework could also be used to implement miniaturized plug and play synthesis modules for previously built interfaces and instruments that comply to specific standards of communication.

Following are links to performances which made use of the rebuilt versions:

- https://youtu.be/U7UMkQxeKC4
- https://youtu.be/6C0c8fTA1eU

7. REFERENCES

[1] L. Brown, S. Brewster, and H. Purchase. A first investigation into the effectiveness of tactons. In *First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 167–176, 2005.

[2] C. Chafe and M. S. O'Modhrain. Musical muscle memory and the haptic display of performance nuance. In *Proceedings of the International Computer Music Conference*, pages 428–431, 1996.

[3] S. Fels, A. Gadd, and A. Mulder. Mapping transparency through metaphor: towards more expressive musical instruments. In *Organised Sound,7(2)*, pages 109–126. Cambridge University Press, 2002.

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

[5] I. Franco and M. M. Wanderley. Prynth: A framework for self-contained digital music instruments. In *Bridging People and Sound*, pages 357–370. Springer International Publishing, 2017.

[6] I. Hattwick, I. Franco, and M. M. Wanderley. The vibropixels: a scalable wireless tactile display system. In *International Conference on Human Interface and the Management of Information: Information, Knowledge and Interaction Design*, pages 517–528. Springer, 2017.

[7] L. Hayes and C. Michalakos. Imposing a networked vibrotactile communication system for improvisational suggestion. *Organised Sound*, 17(1):36–44, 2012.

[8] A. Hunt and R. Kirk. Mapping strategies for musical performance. In *Trends in Gestural Control of Music, M.M. Wanderley and M. Battier*, pages 231–258. Ircam - Centre Pompidou, 2000.

[9] J. Malloch, S. Sinclair, and M. M. Wanderley. Libmapper:(a library for connecting things). In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 3087–3090. ACM, 2013.

[10] 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 66–69, New York City, NY, United States, 2007.

[11] M. Marier. The sponge a flexible interface. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 356–359, Sydney, Australia, 2010.

[12] M. T. Marshall and M. M. Wanderley. Vibrotactile feedback in digital musical instruments. In *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*, pages 226–229, Paris, France, 2006.

[13] A. P. McPherson, E. Berdahl, M. J. Lyons, A. R. Jensenius, I. I. Bukvic, and A. Knudsen. NIMEhub: Toward a Repository for Sharing and Archiving Instrument Designs. Brisbane, Australia, 2016.

[14] E. R. Miranda and M. M. Wanderley. New digital musical instruments: control and interaction beyond the keyboard. In *Computer Music and Digital Audio Series 21*. AR Editions, Inc., 2006.

[15] A. Momeni and C. Henry. Dynamic independent mapping layers for concurrent control of audio and video synthesis. In *Computer Music Journal,30(1)*. MIT Press, 2006.

[16] F. Morreale and A. P. McPherson. Design for longevity: Ongoing use of instruments from NIME 2010-14. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Copenhagen, Denmark, 2017.

[17] G. Torre, K. Andersen, and F. Balde. The hands: The making of a digital musical instrument. In *Computer Music Journal,40(2)*, 2016.