An embedded wavetable synthesizer for the electronic bandoneon with parameter mappings based on acoustical measurements

Juan M. Ramos
Universidad Nacional de Quilmes
Roque Sáenz Peña 352
Bernal, Argentina
juan.ramos@unq.edu.ar

Esteban R. Calcagno
Universidad Nacional de Quilmes
Roque Sáenz Peña 352
Bernal, Argentina
ecalcagno@unq.edu.ar

Pablo E. Riera
Instituto de Ciencias de la Computación (UBA)
Pabellón Cero+Infinito – Ciudad Universitaria
priera@dc.uba.ar

ABSTRACT

The bandoneon is a free-reed instrument of great cultural value that is currently struggling to ensure its conservation as heritage, mainly due to its complex constitution, the lack of sufficient manufacturers to satisfy the demand, and the high sales prices that this entails. Our research group has been working on the task of revitalizing the instrument from a modern perspective, carrying out musical and scientific research for the creation of an accessible electronic bandoneon. As the next step in this endeavor, we present a method for synthesizing the bandoneon sound using multiple wavetable interpolation, and parameter mappings based on acoustic measurements. We discuss a method for capturing and selecting the wavetables, the implementation on an embedded platform (Bela Mini), and the trade-offs between realistic sound and computational efficiency. The synthesizer runs in real-time and has a polyphony of approximately 12 voices, allowing for an autonomously sounding electronic instrument.

Author Keywords

Bandoneon, Synthesizer, Embedded, Wavetable

CCS Concepts

•**Applied computing** → **Sound and music computing;** Performing arts; •**Information systems** → *Music retrieval;*

1. INTRODUCTION

The bandoneon is a free-reed instrument which evolved from the European accordions and concertinas of the 19^{th} century [1, 2]. It is powered by a square-shaped bellows with wooden lids and a keyboard on each side. The left keyboard features bass notes, while the right keyboard has treble notes, intended to be played with each hand on its corresponding side.

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

This work is part of *Bandoneón 2.0*, an interdisciplinary project based in Argentina whose primary goal is to produce electronic bandoneons, and to conduct the academic research required to accomplish this task. A journey through the history and current development of the project can be found in [3]. One of the specific goals of the project is to develop a realistic bandoneon-sounding synthesizer, designed to work in tandem with the electronic bandoneon controller. Let us recall the words of Franco and Wanderley, who raised perhaps the most relevant question regarding the importance of accomplishing this union:

> *How can these new instruments provide the same gratifying immediacy and musical expressiveness as their acoustic counterparts?* [4]

Furthermore, providing the instrument with its own portable (or built-in) synthesizer is key to allowing it to be independent of a computer for generating its sound, making it suitable for use not only in experimental performances but also in more traditional contexts, where bringing even a small computer is not usually possible. As Mulshine and Snyder state, embedded audio synthesis allows for the creation of instruments that have distinctive and long-lasting identities and, at the same time, avoids maintenance issues associated with connection to personal computers [5].

In South America, there have been some attempts to produce electronic bandoneons as detailed in [3]; all of them were developed as MIDI controllers and typically used software samplers running on associated computers to produce their actual sound. As realistic as a sample of a note can be, it is very difficult for it to reproduce the evolving dynamic details and nuances of such an expressive instrument in real time, at least without a thorough understanding of the acoustics of the instrument. To the best of our knowledge, the only closely related instrument with its own specifically designed internal synthesizer is the *V-Accordion* series produced by Roland. In previous work [6], we explored the concept of an embedded FM bandoneon synthesizer designed in the FAUST language and implemented on a Teensy 4.0 board. The choice of its FM parameters and other mappings was based on preliminary acoustic measurements of the bandoneon, which served as the basis (and were extended) for this work.

1.1 Embedded synthesis in NIMEs

Embedded synthesis or DSP systems have empowered many NIMEs. As mentioned by Franco and Wanderley, the first attempts were aimed at creating self-sufficient DSP units [7, 8, 9, 10, 11, 12]. Since then, many platforms have been used to this end, particularly ARM based devices. Among

these, the Beaglebone Black (released in 2013) is a powerful and low-cost ARM based board capable of running a Linux OS. Several NIMEs used this platform, such as Range [13] and El-Lamellophone [14]. Franco and Wanderley even conducted comparative performance tests on the Beaglebone Black, and found that the wavetable synthesis method was capable of sustaining more than 180 voices, while other methods such as granular or additive reached less than 30.

Based on the Beaglebone hardware, Bela [15] is an open-source platform for sensor and audio processing, which runs a custom real-time audio environment based on Xenomai Linux. Bela has been used in many NIME projects. Moro et al. [16] showed how it can be integrated with Pure Data to perform audio signal processing in real-time. Martin et al. [17] used Bela to interface with Myo sensors in order to control a bank of oscillators with the performer's muscle gestures. Bela has also been used in augmented instruments, such as the work by Gonzalez Sanchez et al., who describes the design and construction of a collection of Bela-equipped augmented acoustic guitars [18]. Pardue et al. created a hybrid digital-acoustic violin [19], which uses a Bela Mini installed in the violin itself to process the string signals in real-time, and acoustically feed the results to the body of the instrument through an actuator.

Besides the Beaglebone/Bela, there have been other developments that make use of platforms such as Teensy, Raspberry Pi, ESP32, STM32 and others [5, 20, 21, 3]. In the following sections, we will describe the electronic bandoneon controller called "Alfa" [22], the wavetable bandoneon synthesizer, and its embedded implementation on a Bela Mini.

2. THE ACOUSTIC AND THE ELECTRONIC BANDONEON

An acoustic bandoneon has 71 keys (typically), each one producing a different note depending on whether the bellows is expanded or contracted. Though variations exist, its usual total range is from C2 to C7. Also, most notes -with the exception of the highest pitched ones- are actually produced by two paired reeds tuned to the expected fundamental plus its octave respectively, giving the instrument a very distinctive timbre. The pressure applied by the bellows to the reeds governs the timbral characteristics and the instantaneous pitch of the reeds. Typically, the pitch lowers as the pressure increases for notes up to the highest octaves, where this tendency gets reversed. The attack and release times vary with the note due to their different masses and, to a lesser extent, with the initial pressure. Additional nuances of sound production depend on factors such as the key pressing techniques or timings and its relationship with the bellows' gestures.

The electronic bandoneon we developed, called "Alfa", functions as a controller interface, and has no actual reeds, so its sound has to be generated externally by an appropriate synthesizer. Besides the absence of reeds, the controller has the same general structure as an acoustic bandoneon, including the bellows and number of keys. It senses the air pressure inside the bellows, treating it as a differential amount with respect to the external ambient pressure, so it can be a negative or positive value when expanding or contracting the bellows. The keys states are regularly monitored and, in combination with the pressure value and sign, the appropriate control messages are produced. Any MIDI enabled synthesizer can be used with the controller but, typically, it is expected that the value of the pressure would control the loudness, timbre, and pitch of the sound, while the keystrokes define the attack and release events. Alfa is based on an Arduino microcontroller which manages the sensors, senses the keys and sends the appropriate MIDI messages, with pressure information encoded as CC messages and notes as note-on/off mapped to channel 1 or 2 for the right and left keyboard respectively.

Figure 1: "Alfa", the electronic bandoneon.

3. WAVETABLE SYNTHESIZER

Multiple wavetable synthesis [23, 24] is an additive synthesis technique based on the addition of fixed waveforms or periodic-based functions with variable weights over time.

Unlike classical additive synthesis, where the waveforms to be added are typically sinusoids, wavetable synthesis loads each table with one cycle of a waveform of arbitrary complexity [25]. The main advantage of this technique is its computational efficiency since the number of wavetables used is usually much less than the number of sine waves that would be used in classical additive synthesis [26]. To capture the evolving spectral characteristics of a sound, the basic technique uses several breakpoint times to take snapshots of the waveform and then generates the evolution of the full waveform interpolating between tables from adjacent breakpoints. Other methods separate and mix regions of the spectrum instead of the raw cycles [27], or compute just the differences between breakpoints. There are many strategies to select the breakpoints to be used; in our particular case, we know that the bellows' pressure is the parameter that dominates the sound characteristics for a given note: the more pressure, the more intensity, brightness, detuning, and other spectral features.

In the next section, we will elaborate on the recording and analysis of audio samples and the method for breakpoint selection.

3.1 Samples recording and selection

In order to implement the wavetable synthesis method, we first need to record samples from a bandoneon. To this end, we used an old Uhlig brand bandoneon, which we repaired and utilize for exploratory studies without running the risk of damaging a better-shaped or borrowed professional instrument. It was recorded using our Integrated Measurement System [28], thus obtaining samples from all the reeds, carefully separating fundamentals from octaves.

This was done because the relationship between these reeds is not a perfect octave, and their combined sound may not be perfectly harmonic. For each reed, we executed a slow crescendo-decrescendo, from the quietest to the loudest dynamics. The system allows for capturing the sound and the pressure data synchronously. The resulting waveform is exemplified in figure 2 along with the bellows' pressure for an F#4 note. This execution methodology allowed us to capture all the dynamics in a single take. The crescendo and decrescendo regions go through the same overall dynamics, but there are subtle differences due to the rigidity of the bellows when the movement begins (i.e. increasing the force) compared to when it ends (releasing the force). After a careful inspection and comparing it with professional instruments, the captured audio was equalized to compensate for some of the high-end loss of our old instrument due to its age and condition, but also for acoustic conditions that produced a slightly exaggerated low-end.

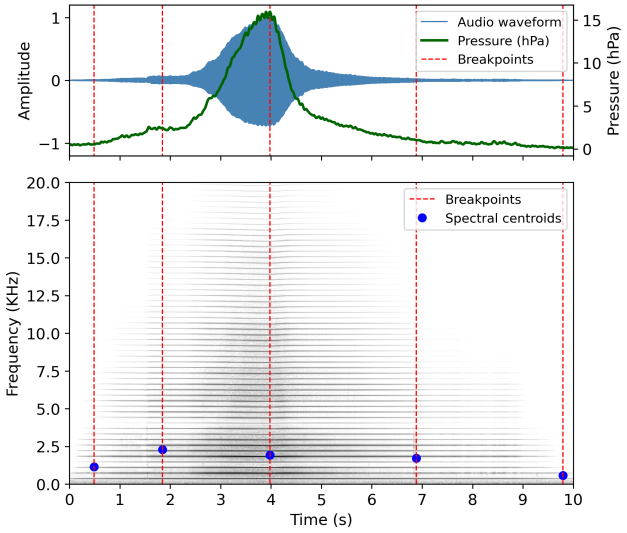

Figure 2: Top: Single reed recording with a crescendo-decrescendo dynamics. The pressure signal is depicted on top of the waveform. The red vertical lines indicate the breakpoint times used to select the wavetables, which waveforms are shown in figure 3. Bottom: Spectrogram of the recording. The breakpoint times are selected based on the spectral centroid values.

The next step for constructing the wavetables is to select a number of waveform segment candidates distributed linearly along the whole take, including a segment in the intensity peak. We performed a DFT analysis over the segment and measured the phases and amplitudes of each partial. This allowed us to overcome the potential inharmonicity of the waveforms. Then we performed a resynthesis with every partial as an integer multiple of the fundamental frequency, thus obtaining a harmonic sound with an arbitrary table-size and partial count for each note, which is useful for the implementation in code as discrete points. This resynthesis generates a clean sound, discarding the background noise from the bellows blowing air in the original recording, which will be added separately. For each note, the partials' phase relationship of the cycles corresponding to the different intensities must be the same in order to prevent artifacts when morphing between them. For this, and within any given note, we selected the phase profile of the cycle with the highest intensity and applied it to the rest. We also tried to use constant, random or even ignoring the phases as suggested by some authors [27], but we found that, for this instrument, it was essential to preserve them as much as possible. After some testing, we decided to use 4x oversampling on each wavetable, this is further explained in the next section. Finally, each cycle was labeled with its associated pressure, RMS and precise frequency values, as extracted from the initial analysis.

The wavetable cycles obtained (typically 50 for each reed) are now candidates for being used in the synthesizer as breakpoints for interpolation. In the examples presented in this work, we used a limited set of 5 breakpoints for each note, as we found it sufficient to represent a smooth transition from lowest to highest intensity. To select the best representatives, we first picked the wavetable at the peak intensity, and then we selected four more wavetables using a spectral criterion, taking a linear range of spectral centroid values. In figure 2, the vertical lines indicate the center of the segments selected as breakpoints. In the spectrogram, we can see how the spectral characteristics evolve and change with the pressure exerted. Is interesting to note the increase in background wind noise when the pressure is higher. In the final stage, the wavetables are exported along with their labels for pressure and RMS values. In figure 3, we can see the corresponding 5 selected waveforms for the note in figure 2.

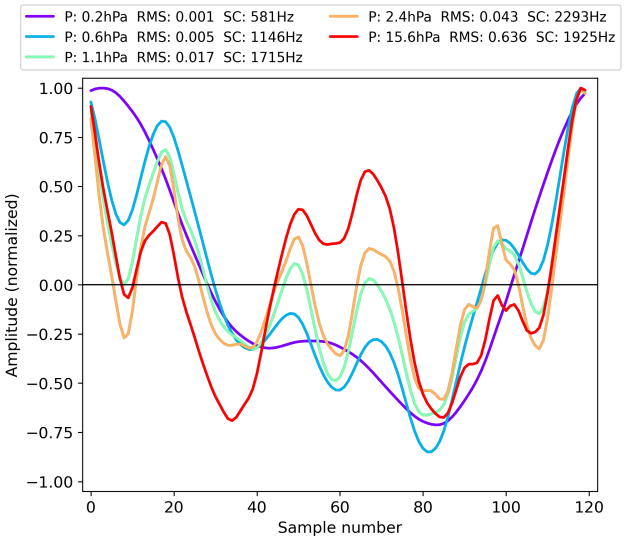

Figure 3: 5 amplitude normalized wavetables for the recording example in figure 2. For each one, we show the bellows pressure, the RMS amplitude at the recording, and the spectral centroid value.

All the process was automatized, but each breakpoint selection was manually inspected to check for errors. In future work, we will expand the use of other spectral characteristics to look up a fully automated procedure.

3.2 Synthesis

The complete synthesizer has the following parts: the primary audio signal is generated by separate wavetables for each of the two reeds (fundamental and octave). The pressure signal controls the fundamental frequency and amplitude, while the keys (left or right hand) select the notes and trigger an amplitude envelope. Finally, two more audio signals are added with wind noise and keystroke sounds.

The primary signal is generated with a standard wavetable interpolation. As the bellow's pressure changes, linear interpolation of the wavetables between two adjacent breakpoints occurs, except in the extreme breakpoints (quiet and

loud) where no interpolation occurs. Time-adjacent samples are also interpolated to allow for real-time pitch control, and we use 4x oversampling to reduce aliasing while maintaining linear interpolation. Higher order interpolation methods can be used, but have a higher CPU cost. The starting phase of each signal is randomly selected to avoid exactly repeating sounds when summing the fundamental and the octave.

Each key has a note which assigns the corresponding base fundamental frequency, with some noise in its value to make the whole instrument sound more natural. The octaves are also slightly detuned to this effect. On top of the fundamental base frequency, some detuning occurs at high pressures. Figure 4 shows the pitch shift from the expected note for different pressures. Each dot represents a point in the crescendo-decrescendo waveforms for all the recorded reeds. We indicate with lines the evolution of the detuning for representative notes in each octave. We can see that for low octaves, the detuning can be very pronounced, almost half a semitone; however pressure extremes are seldom used while playing. The mapping of pressure to pitch shifting is characterized by performing a linear regression for each note.

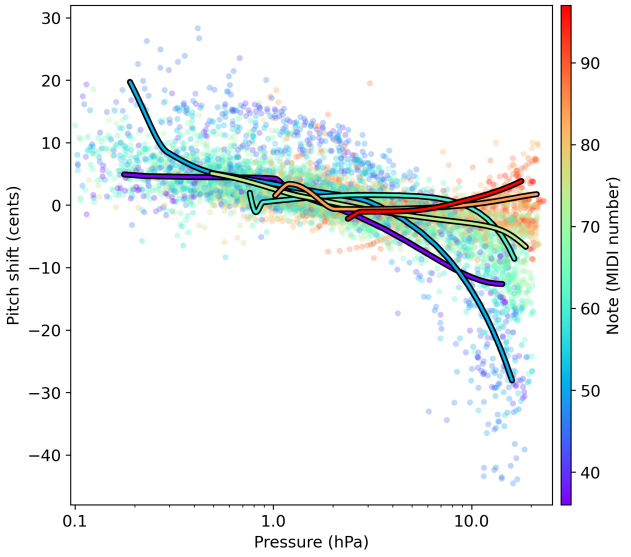

Figure 4: Detuning of reeds as pressure increases. Each dot represents a cycle taken from recordings of all reeds. The pitch shift is measured relative to the mean reed pitch. We show six lines corresponding to notes spanning 6 octaves to better illustrate the effect that lower notes get a lower pitch with high pressure, while higher notes do the opposite.

The instrument has two keyboards with some notes overlapping across both, but having distinct spectral characteristics, mainly due to a resonator structure present on the left side. The synthesizer takes this into account, so each hand has its own wavetable set created with its corresponding recordings, and a panned stereo signal can be generated.

The fine temporal dynamics when a key is pressed are modulated by an attack-sustain-release amplitude envelope. On top of that, the overall level is continuously modulated by the pressure with linear interpolation of the RMS amplitude corresponding to each breakpoint.

The attack time of the envelope varies with the note and the pressure, following the formula $attack\_time \propto \epsilon(-\alpha n + \beta)(PMax - \gamma p)$ where n is the note and p the pressure. The release time formula is similar, but it does not depend on the pressure. We adjusted the parameters to obtain times values based on our own measurements [6]. The attack and release transients are modeled with the formula $a(t) = sin(\pi t^2/2),\ t \in (0, 1)$. Where t is normalized to be one at the attack time. These transients simulate the response of a high-order filter that the reed's onset would follow in response to pressure changes. From previous work [3], we know the attack time gets shorter with frequency so, in the synthesizer, we need to make the envelope of the octave slightly faster. To save computation time, we implemented a strategy to reuse the fundamental's envelope on the octave reed, by multiplying its value by 1.5 and clipping the result to 1, which produces a steeper slope and a faster transition. Another significant peculiarity of the reeds is their remaining energy and oscillation after the key has been released, which is typically very quiet and is ignored; since air cannot flow anymore, it behaves more like an idiophone. However, this effect greatly impacts the reed's ability to begin sounding again after a new keypress, significantly reducing its attack time if any oscillation remains. This affects the ability to perform quick *trinos* and repetitions, which wouldn't be possible if the note always starts off as if it had zero residual oscillation. We simulated this effect by adding a counter that acts as a hidden (and slightly longer) release phase, which is checked at every note-on event, and scales down the attack time if it has a non-zero value.

In figure 5 we show the spectrograms of the original recording on the left (same signal as in figure 2) and the synthesized with the wavetables using the same pressure profile on the right.

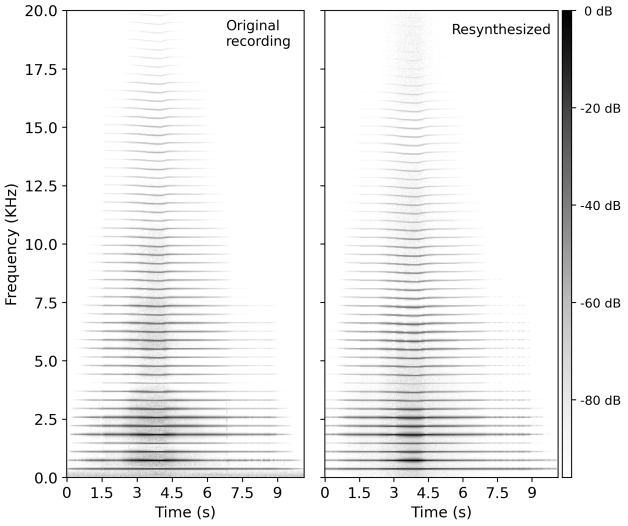

Figure 5: Comparison of spectrograms of the original recording and the synthesized one with wavetables. Stronger detuning can be observed in the high pressure regime. All the partials have the same relative detuning (i.e. the signal is harmonic), but the linear frequency scale shows the deviation more prominently in the higher ones.

Finally, the synthesizer produces two more kinds of sounds in parallel with the reeds: keystrokes and wind noises. A limited set of recorded keystroke samples were added to the system, which are played at each note-on and note-off event. Each keystroke sample is assigned to a specific note so each note has its own sound. The set of samples is distributed in a cyclic way using the note number. As a final result, the order appears randomly distributed to the ear while playing. We also added a filtered noise generator in order to simulate the wind noise produced when there's high pressure is applied to the reeds. This is particularly notorious both on

very fast and loud attacks (there's even a common staccato technique called *escupido* that depends on this effect) and also at any loud sustain in general, as can be seen in figure 2.

All these elements are available as modifiable parameters, allowing us to change from small details to more fundamental features and, for example, simulate other instruments such as an accordion. In the next section, we discuss the implementation of the synthesizer in the embedded system.

4. EMBEDDED IMPLEMENTATION

The Bela Mini is a powerful low-cost microcomputer based on the PocketBeagle, which features a 1 GHz ARM Cortex-A8 CPU, 512MB of DDR3 RAM, and runs a Xenomai Linux OS that manages the interfacing with external devices (such as MIDI controllers), provides onboard code compiling and many other user-friendly features. Bela has 8 channels of 16-bit analog input and output, and 16 digital I/O at audio rate and is capable of audio latencies as low as 80 microseconds in specific configurations [15].

Regarding memory usage, we found that 512MB was enough for the case of a single-instrument wavetable synthesizer, even accounting for Xenomai's overhead. In its current implementation, we measured the physical memory footprint of the synthesizer at approximately 22MB (less than 5% of the total). Besides the wavetables themselves, we stored many other tables with constants like the pressure and RMS mappings, and also many precalculated values to speed up the audio render loop. Despite having more available memory than needed, we made some memory-saving decisions, such as storing only half of the notes (reusing the nearest tables for the remaining notes) in order to test the tolerances of our design empirically.

CPU-wise, we successfully obtained a maximum polyphony of 10-12 simultaneous voices (each voice being a complete note with both reeds, wind noise, and keystroke sounds), with a block size of 256 and the CPU usage peaking at 90%, where it began to drop frames or even crashing sometimes. In this sense, we estimated 6 voices as the bare minimum necessary for a bandoneon, but around 8-10 as a safer margin to allow the release tails to decay correctly. Smaller block sizes worked with reduced or unstable polyphony: about 8-10 voices with block sizes 16-128, while no noticeable improvements were found above 256, which seems to suggest a bottleneck after 8-12 voices. We will continue to explore this issue for the next iteration of our synthesizer, but for our current purposes, the polyphony achieved was enough to properly test the overall design.

In order to achieve this performance, we had to take several optimization strategies in the actual implementation, and even compromise some of the final realism aspects by simplifying operations and mappings. As mentioned, linear equations were used to map attack and release times -which are only computed at note-on events-, and faster interpolated lookup tables for sample-to-sample mappings like the amplitude and pitch changes. Where possible, we made use of the Math-Neon [29] library to obtain less accurate but faster functions such as *sinf_neon* (sine) to map the envelope with a sinusoidal shape. Random values for the wind noise are generated by means of a linear congruential generator algorithm every four samples, which produces a subsampling both to reduce computations and to make it sound warmer than plain white noise. There is also a small lookup table of 32 random values to set an initial phase for each reed at the note on events (necessary to have a more natural phase heterogeneity). We profiled these optimizations in hardware by setting a GPIO pin high and low, allowing us to measure code timings with an oscilloscope and compare the performance of different strategies.

Regarding the MIDI inputs from the electronic bandoneon, we had to adapt both the controller and the synthesizer to compensate for the low resolution of the 7-bit MIDI CC messages for the pressure signal. Although 14-bit CC can be easily implemented on the Bela, the Arduino microcontroller in Alfa can't process data and send double the amount of messages per unit time (two as opposed to one 7-bit message) fast enough, and it tends to miss critical events. To overcome this, we compressed the pressure signal by applying a cube-root mapping in the controller, which provides better effective resolution in the -far more common to use- low-pressure range. The Bela then decompresses the signal by cubing the input. We also implemented a MIDI-thru in the Bela, allowing us to send a copy of the received messages to a computer and record the stream. In the next version of our controller we'll be looking for a direct integration with the synth platform that could allow for more accurate keystroke and air pressure sensing (and messaging), making MIDI available externally but not required internally.

The controller is connected to the Bela by USB, and *ideally* the whole setup is powered by batteries. However, when working on the code and testing the synthesizer (or recording MIDI and audio), Bela was connected to a computer both by USB and to the sound interface. This produced a lot of noise in its output, probably due to ground loop issues. We mitigated this effect by using a filter for car audio applications between Bela and the sound interface. No such noise was observed when powering Bela with batteries.

5. PLAYING RESULTS

After our initial assessments, we brought a professional musician to test the synthesizer with our Alfa bandoneon controller. We recorded the audio, video, and MIDI stream of various songs and practice techniques in order to have a palette of sounds and expressions, both to conduct real-time parameter adjustments and to collect data for further analysis. We immediately repeated and recorded these performances with an acoustic bandoneon, to have a point of comparison that would allow us to make relevant modifications to the synthesizer. Collecting the MIDI streams of these performances was extremely useful, as allowed us to set up an iterative mechanic to tune various parameters of the synthesizer and to adjust the breakpoint selection algorithms to find a better sound match than what was achieved during the original recordings. A selection of this material is available at our website `https://bandoneon.ar/publicaciones`.

The overall experience was very good, and praised as capable of much more expressiveness than sampler methods, although the timbre of some regions still needs some work. In this regard, and having tested our method, we intend to record a new set of samples from a professional instrument.

We also experimented with tuning some parameters outside their typical ranges. An accordion-like sound was relatively easily achieved by making the octave reeds no longer octaves but detuned unisons of the fundamentals. Harmonica, melodica, and even one-voice reed organ sounds can be achieved by discarding the second reed entirely and allowing for heavier note bending with overpressure in the case of the harmonica (though this proved difficult to control without a more traditional interface such as a pitch wheel). Electronic piano sounds were achieved by setting faster attack times and distorting the breakpoint search algorithm to give less bright breakpoints more prominence.

6. CONCLUSIONS

In this article, we presented our progress in creating a fully portable and expressive electronic bandoneon, through the development of an embedded synthesizer based on the technique of multiple interpolated wavetables. Our method for capturing and selecting the wavetables has proven to be robust and sufficient for this stage of the research, although we will continue to look for ways to improve automatic breakpoint detection.

The synthesizer has numerous parameter mappings based on our analysis of the acoustic behavior of the bandoneon. We found that, for a given note, the parameter that best describes the behavior of the instrument is the pressure exerted by the bellows. We have managed to reproduce effects such as timbre, intensity, and pitch changes based on pressure. We also incorporated the sound of keystrokes and the noise of the wind flowing through the various reeds at high pressures.

Our implementation in a Bela Mini has achieved a polyphony of up to 12 voices, although we will continue working on optimizing its performance. We have successfully tested the synthesizer with our "Alfa" electronic bandoneon controller and produced audio, video, and MIDI material for further analysis and iterative improvements.

7. ACKNOWLEDGMENTS

We'd like to thank our families, Universidad Nacional de Quilmes, and Joaquin Rizza, for their constant support. We'd also like to thank the NIME community, the Bela team, and Manuel Eguía for their support on obtaining the Bela Mini that made this work possible.

8. ETHICAL STANDARDS

This work is currently being carried out with financial support from Consejo Nacional de Investigación Científicas y Técnicas (CONICET, Argentina), Universidad Nacional de Quilmes (Argentina), and Ministerio de Cultura (Argentina). The authors take special care in their ongoing research to respect the cultural and historical background of the bandoneon and its importance to their region, and not only abide by applicable laws about the protection of the instrument (such as Argentinean Law 26,531), but also encourage its dissemination and compliance within and outside the national territory. In this sense, the *Bandoneón 2.0* project is in no way an effort to "replace" the acoustic bandoneon, but to help preserve it by making it easier for everyone to access it and increase their interest in the instrument in general.

9. REFERENCES

[1] S. A. Eydmann. The life and times of the concertina: The adoption and usages of a novel musical instrument with particular reference to scotland. *The Open University.*, 1996.

[2] M. Krapovickas. Organografía del bandoneón y prácticas musicales: Lógica dispositiva de los teclados del bandoneón rheinische tonlage 38/33 y la escritura ideográfica. *Latin American Music Review*, 2012.

[3] Juan Ramos, Esteban Ramón Calcagno, Ramiro Oscar Vergara, Pablo Riera, and Joaquín Rizza. Bandoneon 2.0: an interdisciplinary project for research and development of electronic bandoneons in Argentina. In *NIME 2022*, jun 16 2022. https://nime.pubpub.org/pub/31l4lgcd.

[4] Ivan Franco and Marcelo M Wanderley. Practical evaluation of synthesis performance on the beaglebone black. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 223–226, 2015.

[5] Michael Mulshine and Jeff Snyder. Oops: an audio synthesis library in c for embedded (and other) applications. In *NIME*, pages 460–463, 2017.

[6] Juan Ramos, Esteban Calcagno, Ramiro Vergara, Joaquín Rizza, and Pablo Riera. An electronic bandoneon with a dynamic sound synthesis system based on measured acoustic parameters. *Computer Music Journal*, 46(1-2):1–18, 2023.

[7] Edgar Berdahl and Wendy Ju. Satellite ccrma: A musical interaction and sound synthesis platform. In *NIME*, pages 173–178, 2011.

[8] Chris Carlson, Eli Marschner, and Hunter McCurry. The sound flinger: A haptic spatializer. In *NIME*, pages 138–139. Citeseer, 2011.

[9] Hongchan Choi, John Granzow, and Joel Sadler. The deckle project: A sketch of three sensors. In *NIME*, 2012.

[10] Steve Curtin. The soundlab: a wearable computer music instrument. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 1994.

[11] A. J. Thibodeau Hollinger and M. M. Wanderley. An embedded hardware platform for fungible interfaces. In *Proceedings of conference on New Interfaces for Musical Expression*, 2010.

[12] Sukandar Kartadinata. The gluiph: a nucleus for integrated instruments. In *Proceedings of the 2003 conference on New Interfaces for Musical Expression*, pages 180–183, 2003.

[13] Duncan MacConnell, Shawn Trail, George Tzanetakis, Peter Driessen, Wyatt Page, and N Wellington. Reconfigurable autonomous novel guitar effects (range). In *Proceedings of the international conference on sound and music computing*. Stockholm Sweden, 2013.

[14] Shawn Trail, Duncan MacConnell, Leonardo Jenkins, Jeff Snyder, George Tzanetakis, and Peter F Driessen. El-lamellophone a low-cost, diy, open framework for acoustic lemellophone based hyperinstruments. In *NIME*, pages 537–540, 2014.

[15] Andrew McPherson. Bela: An embedded platform for low-latency feedback control of sound. *The Journal of the Acoustical Society of America*, 141(5):3618–3618, 2017.

[16] Giulio Moro, Astrid Bin, Robert H Jack, Christian Heinrichs, Andrew P McPherson, et al. Making high-performance embedded instruments with bela and pure data. *University of Sussex*, 2016.

[17] Charles Patrick Martin, Alexander Refsum Jensenius, and Jim Torresen. Composing an ensemble standstill work for myo and bela. *arXiv preprint arXiv:2012.02404*, 2020.

[18] Victor Evaristo Gonzalez Sanchez, Charles Patrick Martin, Agata Zelechowska, Kari Anne Vadstensvik Bjerkestrand, Victoria Johnson, and Alexander Refsum Jensenius. Bela-based augmented acoustic guitars for sonic microinteraction. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 324–327. Virginia Tech, 2018.

[19] Laurel Smith Pardue, Kurijn Buys, Michael Edinger, Daniel Overholt, and Andrew McPherson. Separating sound from source: sonic transformation of the violin

through electrodynamic pickups and acoustic actuation. In *NIME 2019 New Interfaces for Musical Expression conference*, pages 278–283, 2019.

[20] Andrew Piepenbrink. Embedded digital shakers: Handheld physical modeling synthesizers. In *NIME*, pages 362–363, 2018.

[21] John Sullivan, Julian Vanasse, Catherine Guastavino, and Marcelo M Wanderley. Reinventing the noisebox: Designing embedded instruments for active musicians. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2020.

[22] Juan Mariano Ramos. Bandoneon 2.0: desarrollo del prototipo de bandoneón electrónico alfa. *UFSCar, editor, Anais das XXVII Jornadas de Jovens Pesquisadores. A Ciência ea Tecnologia na Produçao de Inovaçao e Transformaçao Social., Sao Carlos, Brazil*, 2019.

[23] Andrew Horner, James Beauchamp, and Lippold Haken. Methods for multiple wavetable synthesis of musical instrument tones. *Journal of the Audio Engineering Society*, 41(5):336–356, 1993.

[24] Andrew Horner. Computation and memory tradeoffs with multiple wavetable interpolation. *Journal of the Audio Engineering Society*, 44(6):481–496, 1996.

[25] Robert Bristow-Johnson. Wavetable synthesis 101, a fundamental perspective. In *Audio engineering society convention 101*. Audio Engineering Society, 1996.

[26] Jonathan Mohr and Xiaobo Li. Computational challenges in multiple wavetable interpolation synthesis. In *Computational Science—ICCS 2003: International Conference, Melbourne, Australia and St. Petersburg, Russia, June 2–4, 2003 Proceedings, Part I*, pages 447–456. Springer, 2003.

[27] Andrew Horner. Wavetable matching synthesis of dynamic instruments with genetic algorithms. *Journal of the Audio Engineering Society*, 43(11):916–931, 1995.

[28] Ramos Juan, Esteban Ramón Calcagno, Vergara Ramiro Oscar, Rizza Joaquín, and Pablo Riera. Sistema integral de medición (SIM) de parámetros acústicos para el desarrollo de bandoneones electrónicos. (in press). In *Actas de la AES LAC '22*, 2023.

[29] Math Neon. Math-neon, available at https://code.google.com/archive/p/math-neon/.