Towards the Continuous Harmonium: Replicating the Continuous Keyboard

Travis J. West
travis.west@mail.mcgill.ca
IDMIL, CIRMMT, McGill University
Montreal, Canada
Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189
CRIStAL
Lille, France

Ninad Puranik
CAML, CIRMMT, McGill University
Montreal, Canada

Gary Scavone
CAML, CIRMMT, McGill University
Montreal, Canada

Marcelo M. Wanderley
IDMIL, CIRMMT, McGill University
Montreal, Canada

Abstract

In our effort to develop an augmented harmonium to enable the performance of continuous pitch ornamentation while preserving typical harmonium gestures, we have replicated the continuous keyboard presented by McPherson et al. in prior work. We present 1) our adaptations to the design of the sensing system, 2) our preliminary novel mapping design, and 3) a report on our replication process.

Keywords

Replication, augmented keyboard, continuous keyboard, optical sensing, harmonium, mapping

1 Introduction

The harmonium is keyboard instrument played in classical Hindustani music, where it typically plays a role that supports the vocalist. However, because of its use of the western musical keyboard, the harmonium is unable to recreate the pitch ornaments performed by the vocalist. Harmonium players have developed a variety of techniques that aim to mimic the vocalist, working around the discrete pitches of their instrument[6].

We aim to develop an augmented harmonium that preserves these idiosyncratic gestures, but enables a more faithful recreation of vocal pitch ornamentation. Our initial efforts focused on mappings based on data available from a typical MIDI keyboard, however we quickly realized that the limited information provided by MIDI note messages would not support us in achieving our goal. We opted to look for a low-cost method to acquire more data about harmonium players' gestures when performing their instrument. Chapter 2 of Giulio Moro's 2020 thesis [4] provides a good overview of keyboard sensing strategies. We decided, as a first step, to replicate the optical keyboard sensing system used by Moro, presented in detail by McPherson in 2013 [3]. This system continuously measures the angle of the key, or equivalently the partial depression of the key; we hypothesized that this information would be sufficient to achieve our goals. We present our adaptations to McPherson's system, our preliminary mappings, and a report on our replication effort.

NIME '25, June 24–27, 2025, Canberra, Australia

2 Prior Work

Andrew McPherson's continuous keyboard sensor comprises the following parts: an op-amp conditioning circuit, and a microcontroller for data acquisition and communication with an application processor such as a laptop, and an array of near-field infrared reflection sensors each of which comprises an infrared emitter and phototransistor in one package. Multiple phototransistors can be connected in parallel to one op-amp circuit as long as the infrared emitters can be toggled on and off sequentially; as long as only one emitter is turned on, only its associated phototransistor should be activated, and the measurement from the output of the op-amp should correspond to distance relating to the activated emitter/phototransistor pair.

Although it is not explicitly stated, we inferred from McPherson's description of the system [3] that each infrared emitter is driven directly from one output pin of the microcontroller, via a switching transistor. In this way, each emitter can be switched on independently, enabling quick scans simply by sequentially turning the output pin connected to each switching transistor on and off.

McPherson's system makes use of four circuit boards, each hosting one microcontroller and up to 25 infrared sensors. The boards are connected to a shared serial peripheral interface (SPI) bus. One of the boards acts as manager, polling data from the other boards, maintaining clock synchronization, and communicating with the application processor.

Each sensor is read at a sampling rate of 1 kHz, where each sample is differential and oversampled, with eight measurements of the analog response when no emitters are enabled and eight measurements of the response when the emitter is switched on.

3 Our Adaptations to the Continuous Keyboard

Our first main contribution comes from the changes made in our version of the continuous keyboard sensor with respect to prior implementations. Most notably, unlike previous versions, our implementation uses a matrix configuration that reduces the number of parts in the system. We describe the electronic and firmware design of our implementation.

3.1 Electronic Design

Our version of the electronic design maintains McPherson's op-amp circuit, but otherwise departs from the prior system design.

The first substantive change we made in our implementation was to use the QRD1114 rather than the QRE1113 used in prior work. This change was motivated by two considerations. First, while troubleshooting our first scaled down prototype consisting of a single sensor, our observations suggested that all parts of the system were working except for the phototransistor in the QRE1113, prompting us to try switching to another part we happened to have on hand in order to confirm our hypothesis that the QRE1113 was broken. We switched to a QRD1114 and found that the system worked, apparently confirming our hypothesis. This leads us to argue that the QRE1113 is not appropriate for breadboard prototyping; we believe the phototransistor must have become damaged when inserting the sensor into the breadboard, whereas the QRD1114 was robust against this stress. Considering that the QRD1114 behaved appropriately with the existing conditioning circuit and was also more readily available locally, we opted to move forward with this part in our implementation.

We next opted to use the RP2040 microcontroller on a Raspberry Pi Pico board, since this device was familiar to us as well as being available on hand from prior projects. However, the number of output pins on the RP2040 was too small to individually control the 34 sensors needed to measure every key of our harmonium. This led us to connect the infrared emitters in a matrix configuration (see Figure 2). An LED in the matrix is switched on by driving the pin connected to the anode (long leg) high, providing a source of current, and driving the pin connected to the cathode low, providing a current sink; both pins are configured in output mode, and a resistor in series with the cathode pin limits current through the LED and pins, setting the brightness of the LED and avoiding excess current through the pins.

Using a matrix configuration eliminated the need for 34 switching transistors and current limiting resistors (one each for every infrared emitter), in exchange for requiring 6 current limiting resistors (one for each row in the matrix). Our use of a single microcontroller further eliminated the need for an additional op-amp or op-amps and their associated passive components; all of the phototransistors in our implementation are connected in parallel to one op-amp circuit.

The main trade off of using a matrix is that, since a greater number of sensors are connected to one microcontroller, the overall system sampling rate is necessarily lower than if fewer sensors were connected to multiple microcontrollers making measurements in parallel[1]; we see this as an acceptable trade off at this stage in our design, since a somewhat higher system latency does not prevent us from exploring different mapping strategies. A quantitative comparison of the system sampling rate and other engineering evaluation metrics remains as future work; for now, the system performance is heuristically "good enough" to enable our application of the system to harmonium gesture sensing and facilitating our exploration of different mapping strategies.

4 Firmware Design

We implemented our firmware using Sygaldry [8] as an assembly of four main functional components: the ADC, the sensor scanner, the infrared emitter matrix driver, and the MIDI mapping.

[1]the best-case effective sampling rate of the system f_s, where each sample is the average of N raw measurements, taken at the maximum sampling rate of the microcontroller $f_{\max}$, and where S sensors are read by each microcontroller is given as $f_s = f_{\max}/(SN)$

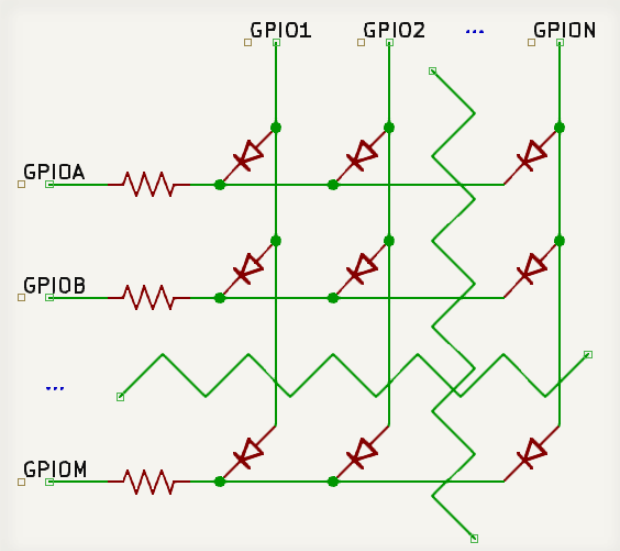

Figure 1: Schematic of the LED matrix circuit; in our implementation there are 6 rows and 6 columns, but different sized matrices would work equally well depending on the number of sensors to be read.

The current version of the firmware can be viewed online[2]; in principle, Sygaldry's design should facilitate future replication of the firmware [8], although this has not yet been demonstrated.

The matrix driver receives the coordinate of the current emitter from the scanner and toggles the output pins of the microcontroller to activate only the associated emitter. The ADC measures the response of the currently activated sensor. The scanner then collects the measurements, associating them with the current key and tracking the minimum and maximum observed values for each key before sending the matrix driver the coordinate of the next key. The MIDI mapping connects the keyboard state to MIDI polyphonic expression values and sends them over USB to a connected host for further application processing, described in the next section.

The ADC component implements a simple oversampling routine. Each output reading is derived from the average of N raw measurements, where N is configurable using a template parameter. Increasing N improves the effective signal-to-noise ratio but reduces the effective sampling rate. Unlike McPherson, we do not measure the ambient light response of the sensors at the time of writing. We reason that since the sensor is installed in the unchanging darkness inside of the harmonium, rather than above the keys as in prior implementations, it should be sufficient to measure the minimum and maximum outputs of the sensors once (e.g. at system start up) in order to adjust the output of the device. Calibration of the non-linear response of the sensor also remains as future work, since the raw output empirically appeared to be reasonably close to linear without calibration.

5 Preliminary Mappings

Our goal is to enable the performance of continuous pitch ornaments characteristic of classical Hindustani music on the harmonium, using the established gestural vocabulary of skilled performers of the instrument as described in [6]. We present

[2]https://github.com/DocSunset/sygaldry/tree/c6080345fd40bcff95dd1c62159c5034a7022e2e/sygaldry-instruments/continulodica_pico

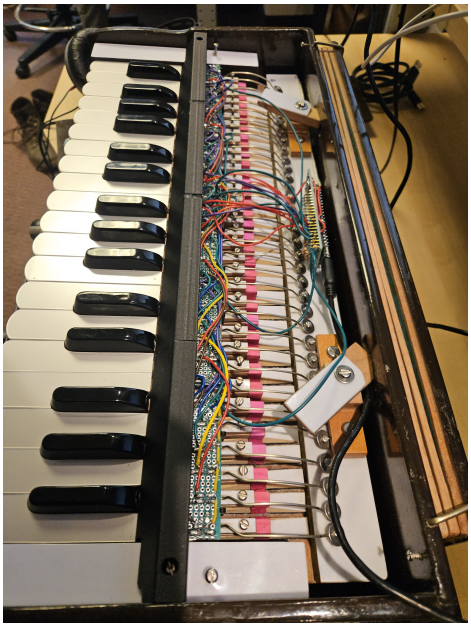

Figure 2: The sensor and microcontroller assembly fits inside the harmonium. It can be covered with the wooden top cover normally present in all harmoniums. This creates a consistent dark environment for the sensors reducing the noise from ambient light. It also maintains the aesthetic look of an acoustic harmonium.

a preliminary mapping of the sensor data to achieve this outcome, making use of a physically informed synthesis model of the harmonium previously described in [5].

Each key is associated with a pitch (i.e. a MIDI note number) and a "weight", which is simply the depth reading for that key. When the weight of a key-press exceeds a small threshold, the key is considered as "pressed". For consistency with an acoustic harmonium, when a single key is pressed, a single chromatic note with the key's associated pitch is synthesized. The amplitude of the synthesized note is proportional to the weight of key-press, with the highest value for a fully pressed key.

For the case when exactly two keys are played in succession in a legato fashion, i.e. when one key is released while another is being pressed, the intended effect of this gesture is to perform a continuous glide between the two notes in consideration. To achieve this, we define a parameter $w' = min(1, w_2/w_1)$ where w_1, w_2 are the weights of the first pressed and the second pressed key, with their corresponding note numbers of p_1 and p_2 respectively. Using the parameter w', the interpolated output pitch p is derived as per the following relation.

$$p = (1 - w') \cdot p_1 + w' \cdot p_2$$

As the glide progresses, w' goes from 0 to 1. At $w' = 1$, the weight w_2 of the second pressed key exceeds w_1, the weight of the previously pressed key. The glide is assumed to be complete when this happens and w' stays at 1 while the previously pressed key is released. Our empirical observations suggest that Hindustani harmonium players rarely press more than 2 keys at the same time. This is because the instrument's role is generally to mimic the monophonic vocals. Pressing of two keys happens only to mimic a crossfade between two notes to create a suggestion of pitch continuity, rather than playing a chord. Hence, the preliminary mapping presented here excludes the cases with three or more pressed keys. This mapping was coupled to a physically informed synthesis model of the harmonium developed by some of the authors [5] to play the music samples presented as supplementary material[3].

6 Replication Report

6.1 Context

Recent discussions have highlighted the benefits of replication [1, 7] and of engaging with old instruments more broadly [2] towards the well-being of NIME practice and research. One major theme in these discussions is the difficulty of documentation; it is challenging to anticipate what information will be most useful to later users and researchers who may wish to replicate an instrument.

When we had finished assembling the circuit described by McPherson [3] and found that it did not work as expected, we were temporarily stymied and became stuck. A very brief discussion with McPherson at NIME 2024 provided all the necessary know how to get unstuck; McPherson advised a series of specific and sequential troubleshooting steps. None of this information was mentioned in the 2013 paper, but it turned out to be very helpful for our successful replication. In the interest of promoting more replication within our research community, we argue that this kind of information, related to challenges and troubleshooting, should be valued, preserved and transmitted.

The replication proceeded through a series of sequential stages: minimal op-amp circuit prototype, small scale sequential prototype, minimal matrix prototype, full-scale prototype, mounting hardware prototypes. From experience, we note that most DMI development work includes such stages, each with their own purpose and procedures, but that this is rarely discussed in the literature, which instead tends to focus solely on the final results of the work. We structure our report to reflect this structure of the development process.

As well as each stage above having its own development and troubleshooting procedures, we note that at each stage we made design choices, following one path of development where another could have been taken. This information is sometimes evident in the discussion of a design, but is often only implied and rarely highlighted. As replication often involves a certain amount of mutation, we expect that explicit presentation of alternatives considered may be especially interesting for future replication efforts.

6.2 Report

Minimal op-amp prototype—**Purpose**: Make sure you understand how the op-amp circuit works, and how it is expected to behave—**Procedure**: Assemble the parts shown in McPherson's schematic [3] on a breadboard, but replace the QRE1113 with the QRD1114. This stage can be done without a microcontroller, using a multimeter to read the output voltage and a fly wire to alternately switch the LED on and off.—**Problems**: Credit to McPherson for the first two troubleshooting tips. Test the op amp circuit by replacing the phototransistor with a potentiometer, enabling you to pull current through the conditioning circuit and make sure that the output voltage varies accordingly. Test the phototransistor by shining a bright light on it. Test the infrared emitter is working using a smartphone camera (most can

[3] https://youtu.be/iFCblP3tDxk

faintly detect infrared light) or an infrared camera. Remember that the long leg of the LED should be connected to the higher voltage.—**Alternative Paths**: McPherson used the QRE1113. We found that ours was broken, possibly from being inserted into the breadboard. We chose to use the QRD1114 instead, since we happened to have one on hand, it worked fine when we put it in the breadboard, and we were able to get more locally.

Small scale sequential prototype—**Purpose**: Make sure you know how the op-amp circuit works with multiple sensors attached, and that you are able to sequentially scan sensors as expected—**Procedure**: Add one or two more sensors to the previous prototype, connect everything to a microcontroller, and start writing firmware—**Problems**: Standard procedure for working out kinks.—**Alternative Paths**: We found we could drive the LEDs without using switching transistors. We opted to omit these components to reduce parts. We realized that the number of pins on our microcontroller would not accommodate the number of sensors needed in the full-scale prototype. We considered using shift registers, but opted instead to use a matrix to reduce parts.

Small scale matrix prototype—**Purpose**: Make sure you understand how to drive LEDs in a matrix configuration. **Procedure**: Connect four sensors with the LEDs in a very small matrix consisting of two rows and two columns. Write firmware (or use the one above) to scan the matrix, oversampling by averaging multiple measurements from each sensor before moving to the next one.—**Problems**: Standard procedures as above. If writing a new firmware, ensure that all pins connected to the matrix are in output mode and driven appropriately. An oscilloscope and/or debug printing can be used to sanity check the behavior of the matrix pins. It can also be helpful to introduce a substantial delay into the matrix scan so that each pin turning on and off can be inspected in real time.—**Alternative Paths**: We accidentally implemented a nonsensical oversampling strategy where the entire matrix was scanned multiple times to produce each oversampled measurement instead of scanning one sensor multiple times in the course of one scan of the matrix. The latter approach is appropriate, since it avoids possible issues related to the response time of the sensors and conditioning circuit. Oversampling is absolutely necessary, since the raw output of the conditioning circuit is too noisy to be used directly.

7 Conclusion

We aim to develop an augmented harmonium that enables the performance of continuous pitch ornaments while preserving the existing gestures of harmonium players. Towards this goal, we developed a new implementation of McPherson's continuous keyboard, which continuously measures the depth of each key press, enabling the development of novel mappings while preserving the traditional interface of the western musical keyboard. In a departure from prior implementations, our implementation makes use of a matrix configuration to scan through the infrared sensors, reducing the number of parts and enabling use of a single microcontroller to scan a greater quantity of sensors. The raw signal is regulated by oversampling and min-max scaling before transmission to the host processor over USB MIDI. Our preliminary continuous pitch mapping is based on the extent of glide progression as determined by the ratio of weights of the two notes played in legato fashion.

Performing continuous pitch expressions on keyboards is challenging. Our novel mapping, combined with a continuous keyboard sensor measuring the depth of key presses, has the potential to enable a faithful performance of such expressions on a keyboard, such as in Hindustani music through the natural gestures of trained acoustic harmonium players. By keeping the mapping simple, we hope to encourage a greater adoption of our interface by existing harmonium players. Apart from the harmonium, the system also opens up the possibilities of playing synthesis models of other continuous pitch monophonic instruments in the Hindustani style. The mapping may also be of interest to musicians performing other styles of music, although our explorations to date have focused on Hindustani music.

As well as presenting the system itself, we offer a brief report on the process of its development based on McPherson's prior system. The replication report highlights the purpose of each stage of development, and the procedures used, problems encountered, and alternative paths considered at each stage, in hopes of aiding subsequent attempts to build a continuous keyboard.

8 Ethical Standards

This work was conducted within the ethical framework of the author's affiliated institutions. We acknowledge the impact on human communities and the environment from the harvesting of rare minerals and other resources involved in the fabrication and movement of electronic components and their constituent materials, as well as the impacts of the infrastructure of the global internet. We have tried to reduce our negative impact by using electronic components already available on hand, and by reducing the number of components required for our implementation.

Acknowledgments

Special thanks to Andrew McPherson for troubleshooting advice.

This work is supported by the CIRMMT student award.

References

[1] Filipe Calegario, João Tragtenberg, Christian Frisson, Eduardo Meneses, Joseph Malloch, Vincent Cusson, and Marcelo M. Wanderley. 2021. Documentation and Replicability in the NIME Community. In *Proceedings of the International Conference on New Interfaces for Musical Expression.*

[2] Raul Masu, Fabio Morreale, and Alexander Refsum Jensenius. 2023. The O in NIME: Reflecting on the Importance of Reusing and Repurposing Old Musical Instruments. In *Proceedings of the International Conference on New Interfaces for Musical Expression.*

[3] Andrew P McPherson. 2013. Portable Measurement and Mapping of Continuous Piano Gesture. In *Proceedings of the International Conference on New Interface for Musical Expression.*

[4] Giulio Moro. 2020. *Beyond key velocity: continuous sensing for expressive control on the Hammond organ and digital keyboards.* Ph. D. Dissertation. Queen Mary University of London.

[5] Ninad Puranik and Gary Scavone. 2023. Physically Inspired Signal Model for Harmonium Sound Synthesis. In *Proceedings of the 26th International Conference on Digital Audio Effects.*

[6] Ninad Puranik, Travis West, Marcelo M Wanderley, and Gary Scavone. 2025. Thoughts on mapping and interface design of a keyboard to perform continuous pitch ornamentations in Hindustani music.. In *Proceedings of the Workshop on Indian Music Analysis and Generative Applications, ICASSP 2025 Hyderabad, India.*

[7] Ajin Tom, Harish Venkatesan, Ivan Franco, and Marcelo M. Wanderley. 2019. Rebuilding and Reinterpreting a Digital Musical Instrument - The Sponge. In *Proceedings of the International Conference on New Interfaces for Musical Expression.*

[8] Travis J West, Marcelo M Wanderley, and Stéphane Huot. 2024. Sygaldry: DMI Components First and Foremost. In *Proceedings of the International Conference on New Interface for Musical Expression.*