The 'E' in QWERTY: Musical Expression with Old Computer Interfaces

Chris Nash
Department of Computer Science and Creative Technologies, UWE Bristol
Frenchay Campus, Coldharbour Lane, Bristol, BS16 1QY, UK
chris.nash@uwe.ac.uk

ABSTRACT
This paper presents a development of the ubiquitous computer keyboard to capture velocity and other continuous musical properties, in order to support more expressive interaction with music software. Building on existing 'virtual piano' utilities, the device is designed to provide a richer mechanism for note entry within predominantly non-realtime editing tasks, in applications where keyboard interaction is a central component of the user experience (score editors, sequencers, DAWs, trackers, live coding), and in which users draw on virtuosities in both music and computing.

In the keyboard, additional hardware combines existing scan code (key press) data with accelerometer readings to create a secondary USB device, using the same cable but visible to software as a separate USB MIDI device aside existing USB HID functionality. This paper presents and evaluates an initial prototype, developed using an Arduino board and inexpensive sensors, and discusses design considerations and test findings in musical applications, drawing on user studies of keyboard-mediated music interaction. Without challenging more established (and expensive) performance devices; significant benefits are demonstrated in notation-mediated interaction, where the user's focus rests with software.

Author Keywords
keyboard input, digital music notation, virtuosity, computer music

ACM Classification
• Human-centered computing~Keyboards • Applied computing~ Sound and music computing • Hardware~Sensors and actuators

1. INTRODUCTION
In digital music, computer and musical keyboards support different aspects of composition and production, which can partition the creative process, between creativity and productivity phases. [11] The embodied interaction style supported by instruments (including MIDI devices) encourages musicians to ideate, compose, and perform music away from the computer, before capturing (recording) it in a realtime performance. In the computer, the sequencer/DAW then draws on keyboard and mouse interaction to edit and refine the musical data, mediated through a visual medium (UI or notation). However, generic computer input styles (keyboard, mouse, WIMP, etc.) are not optimized for fluid and expressive music interaction, splitting and shifting focus between the software environment (notation) and a separate hardware input device (instrument).

The computer keyboard is powerful and precise with respect to symbolic input, but lacks musical fidelity. Data entry is processed as enumerated scan codes, triggered by roughly 100 discrete keys, corresponding to alphanumeric symbols or functions, with modifiers (Shift, Ctrl, etc.) to expand control to additional layers of symbols or functions. Both data entry and program control are designed in sympathy with written language (e.g. mnemonic shortcuts for non-symbolic actions, such as Ctrl-C for *Copy*, or Ctrl-O for *Open*). The keyboard's QWERTY layout enables the development of computing virtuosity, in the form of motor skill and image schemata [12], as seen in touch typing, and similar to the learning mechanisms that allow performers more expressive control of musical instruments. [5] Indeed, the widespread development of such skill among computer users has engendered a design and layout that is resistant to change, and thus we can expect the QWERTY keyboard to be a fixture in studios and workstations for the foreseeable future. However, there is significant scope for adapting and extending the design within the constraints of existing interaction styles to improve expressiveness and control in musical scenarios.

By comparison, musical (MIDI) keyboards are accurate and expressive, but optimized for specialist musical, rather than general-purpose use. Supporting from 25 to 88 discreet, unlabeled, linearly arranged piano keys; the MIDI keyboard's design and ergonomics limit its utility as a computer control device, for tasks other than simple assignable triggers (though [4] offers a fascinating study of the piano as a chording keyboard for text-entry). The MIDI keyboard is designed for expression and creativity rather than efficiency and productivity, which is notably enabled through sensitivity to velocity and pressure (MIDI Velocity and Aftertouch) [1], and enhanced by passive haptic feedback from fully- or semi-weighted keys. In MIDI, pressure is captured at 7-bit resolution (0 to 127), effectively adding a 'continuous' scale of control to the triggering of discrete pitches, significantly increasing the nuance of expressive input, albeit conditional on a concomitant increase in virtuosity.

The goal of this project is to similarly extend the discrete symbolic control offered by the computer keyboard with support for continuous scales of musical control, such as pressure and velocity; increasing the expressiveness of the keyboard without compromising its existing utility, functionality, or ergonomics. Accordingly, the envisaged user experience is characterized by fluid mixed-mode interaction with the device and software: symbolic input and keyboard shortcuts in computer modes, alternating with expressive musical input in performance mode. Specifically; rather than realtime, live musical performance, this research focuses on supporting richer input modes and interaction styles for "offline" notation and music editing, such as those in score editing, sequencing, and DAW software.

Figure 1. A Velocity-sensitive Computer Keyboard.

NIME'16, July 11-15, 2016, Griffith University, Brisbane, Australia.

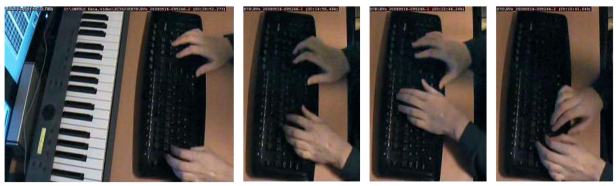
The following sections discuss the background, applications, and development of a prototype device that uses an accelerometer to extend a standard computer keyboard with velocity-sensitivity, as well as other dimensions and properties of musical input. Findings from tests and user evaluations are presented, and discussed in the context of relevant computer music interaction scenarios.

2. BACKGROUND

The *virtual piano* is a common utility provided in music software, or available as a standalone virtual MIDI device (e.g. *VMPK*). A piano keyboard layout is tacitly overlaid on the computer keyboard, allowing pitch to be triggered at a fixed velocity. A degree of polyphony is supported, but due to shared resources in keyboard construction, some key combinations are not possible (the precise limitations vary between keyboard models). Mouse interaction is often also supported, exploiting 2D visualisations to input both pitch (key) and velocity (vertical offset of key click), but cumbersome and imprecise for rapid, accurate entry, even in short phrases. Virtual pianos are thus designed for playing simple melodies, phrases, and monophonic pitch selection, and are designed for convenience in simple tasks (without leaving the computer) or where specialist input hardware is not available (e.g. mobile settings, amateur setups).

While the mouse is an important tool for novice users and unavoidable in some GUI designs, its single point of focus limits the expressive bandwidth and precision of control. In the visual UIs of most professional music editing packages (sequencers, musical typesetters, DAWs), expert users migrate to keyboard shortcuts and bimodal keyboard-and-mouse interaction styles, for more efficient control of the program and direct manipulation styles of editing (e.g. drag and drop). Such literacy with keyboard shortcuts represents a form of computer virtuosity, enabled through repeated practice and learning, significantly improving efficiency and productivity when interacting with programs like *ProTools* and *Logic Pro*. Nonetheless, such packages greatly benefit from specialist hardware input devices for added precision, control, and haptic feedback – MIDI controllers, control surfaces, mixing desks, music controllers, and instruments.

Previous research [8-11] has shown that immersive musical interaction (and creative flow) can be supported in digital creativity, where software maintains user focus, supports rapid edit-audition cycles, and facilitates the development of skilled input and control. This is notably evident in expert use of soundtracking software [8], a text-based music notation manipulated using the computer keyboard, which is used for both musical pitch entry (using a virtual piano) and other properties (via symbolic entry; velocity, volume, and other musical directions), accelerated by a significant number of keyboard shortcuts and editing macros. The notation is manipulated in short editing episodes punctuated by frequent playback of short excerpts, providing rapid musical feedback on edits, lowering the literacy requirement, physically immersing the user in sound, and engendering an iterative, evolutionary approach to crafting music. The rapid oscillation of edits and targeted auditions is enabled through keyboard support, with shortcuts that control playback and efficiently move the cursor focus around the notation. The development of motor skill, including learnt postures and gestures (Figure 2; see [8]), supports a form of embodied music interaction through the standard keyboard. [9]



(a) navigation (b) editing (c) playback (d) piano

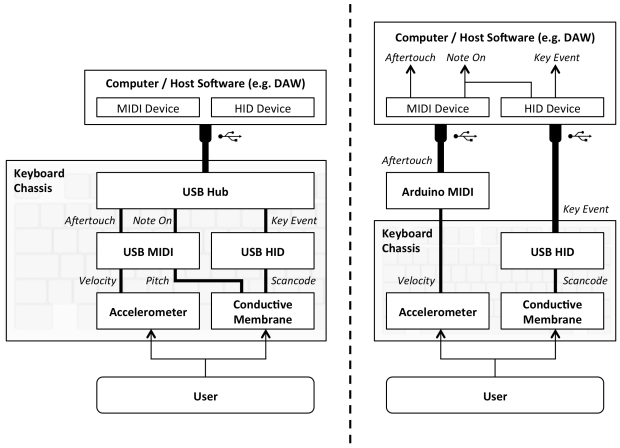
Figure 2. Postures in Keyboard Interaction (see [8]).

This virtuosity, often likened to "musical touch-typing", is widely evident in videos[1] of tracker users (as studied in [8] and [9]), but nonetheless highlights areas for improved ergonomics and expressiveness. For example, the keyboard's lack of velocity control makes auditioning and entering melodic passages limited and cumbersome, requiring manual entry of dynamics. While the virtual piano supports live playback and "key jamming", the fixed velocity offers limited expression, yielding a mechanical performance and hindering the ideation of new musical ideas – consequently, influencing the aesthetic of users with less musical experience. Since dynamics can significantly affect timbre, users may also commit to instrumentations prematurely, not knowing the range of the voice.

While this paper principally explores velocity for musical applications, the additional degree of freedom and analogue input offered by continuous-scale pressure readings have other uses, as explored in other research. Inspired by music keyboards, Microsoft Research previously developed a prototype pressure-sensitive typing keyboard, conceived for use in general-purpose computing [2] – though applications, in gaming (varying walking speed) and instant messaging (mapping 'emotion' to font size), are only briefly discussed and, curiously, no consideration is made of musical use. Their design replaces the standard keyboard's conductive membrane with a custom-manufactured carbon ink printed layer that captures variable pressure, output as modulated voltage (rather than simply closing a circuit). The paper does not detail how the pressures / voltages are integrated with symbolic input or exposed in software or the operating system, nor detail the relative cost and complexity of components and manufacturing. However, the intricacy of the design contrasts the relatively simple, low-cost, and accessible accelerometer augmentation taken in the following section.

3. DESIGN & PROTOTYPING

The keyboard is conceived as a single physical device, presenting two virtual devices to the OS / software: for musical input (MIDI) and computer (data) input, respectively. The USB specification allows both modes to be exposed in software through standard OS drivers, using USB HID and USB MIDI profiles, obviating the need to install additional software or drivers. Using an integrated USB hub, the two devices can be connected using a single USB cable, such that the physical form factor can be identical to the standard computer keyboard. Figure 3(a) shows the architecture of the integrated device, alongside the variation (b) used for testing.



(a) Integrated System (b) Prototype (for testing)

Figure 3. System Architecture.

[1] See https://www.youtube.com/watch?v=SQ5jTaXywuM. [Accessed: 14/04/2016]

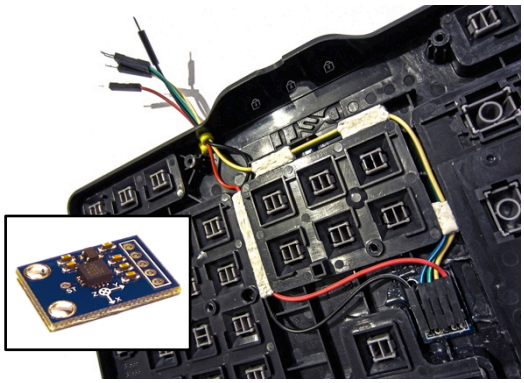


Figure 4. ADXL335 Accelerometer in Keyboard Chassis

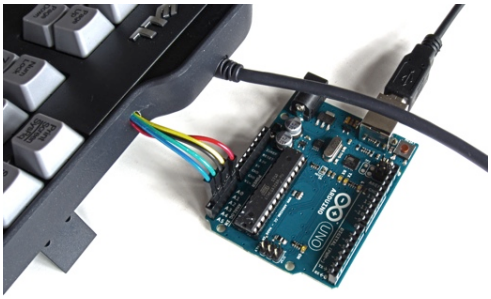


Figure 5. Arduino USB-MIDI Interface[2]

Figure 6. Black and White Piano Key Finish

Figure 7. Arduino-based Prototype

[2] https://github.com/ddiakopoulos/hiduino

The device captures location (key / scan code) data using standard membrane keyboard hardware [2], wherein two plastic layers with conductive strips are separated by a perforated non-conductive layer, such that when pressure closes the gap, a circuit is made. Contact points on each membrane are arranged in a matrix (e.g. one striped vertically, one horizontally) such that each contact can be localized to specific rows and columns, identifying individual keys. A layer of silicon domes is added above the contacts to extend the 'travel' of keys and provide a degree of passive haptic feedback (resistance).

Finger pressure is detected using an accelerometer, secured to the interior of the keyboard chassis (Figure 4). While partially decoupled from the direct impact of individual keys, this arrangement allows a single sensor to be used, and tests demonstrated a good level of sensitivity from conducted vibrations. Other sensor types (such as pressure pads or piezoresistive layers [2]) are also possible, but the accelerometer was chosen for its low relative cost, ease of integration and potential for supporting additional degrees of freedom (e.g. X / Y movement). In the prototype, an Analogue Devices ADXL335 3-axis accelerometer is affixed to a standard Dell keyboard exploiting existing space in the chassis. This low-cost (~$1) sensor has a fixed 10-bit / 3g pressure range, which provided sufficient sensitivity and noise performance for velocity capture (when combined with basic DSP: FIR filtering and peak envelope following), but more sensitive sensors (or averaged multiple sensors, strategically placed in the housing) could offer improved precision.

The accelerometer reading alone can be used to support MIDI *Channel Aftertouch* functionality, allowing software to respond to pressure, or integrated with HID keyboard input in software to create note events with pitch and velocity. To support hardware MIDI *Note On*, the USB MIDI component also requires access to the key / scan code (i.e. pitch / note number). This could be achieved using a *microcontroller unit (MCU)* that detects / sniffs the signals from the keyboard's hardware to identify pitch, either intercepting the encoded serial output of the keyboard's HID chip or scanning the keyboard matrix directly (without compromising HID functionality).

For testing, the prototype model uses an *Arduino Uno* MCU (Atmel ATmega328P; Figure 4), flashed with USB MIDI firmware (using HIDUINO), to create a MIDI device that streams *Channel Aftertouch* messages, where keyboard HID input is subsequently used to detect and generate note events in the computer. This requires additional logic in client software (e.g. the sequencer / DAW) to support *MIDI Note On/Off* events from the setup. The code looks for local maxima in the stream of *MIDI Aftertouch* messages, which are married with corresponding key up/down events that the trigger note events. These events are not synchronous, so data values are buffered until both pitch and velocity are detected. Moreover, testing showed that the order of peak pressure and key press events is not guaranteed – the physical construction of the key mechanism (i.e. haptic resistance provided by the silicone domes) means certain (e.g. soft) touches peak in pressure before closing the circuit, whilst other (e.g. hard) touches peak after. This adds additional complexity to the code, which is encapsulated in a simple C library extension (single header file) – but establishes the methods, in portable code, for embedding in a subsequent integrated hardware device.

Nonetheless, the prototype's simplified architecture (Figure 3b) and software layer means it is not natively supported in existing MIDI programs. However, this avoids one complication of the dual mode design, where conventional text entry will generate MIDI notes for any listening music software (even running in the background). In the integrated device, this necessitates a toggle for MIDI functionality, which could be a hardware switch, hardcoded keyboard shortcut, or one of the existing, seldom-used lock toggles (e.g. *Scroll Lock*). Alternatively, to avoid any interference with HID functionality, it would also be possible to use the accelerometer to detect orientation or gestures as a method for toggling the mode (or changing other settings). Nonetheless, in all key functional respects, the prototype (Figure 7) performs as the final model would.

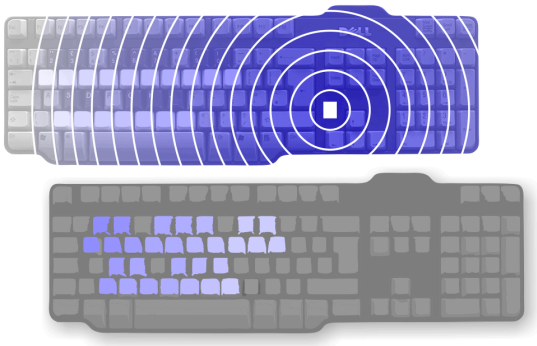


Figure 8. Pressure Sensitivity (top) and Compensation Map (bottom).

To emphasize the musical affordances of the device and to help new users adapt to the virtual piano layout, keys are colour-coded in white, black, and grey (Figure 8). The prototype model achieves this by amalgamating interchangeable key caps from two keyboards of similar model, grey and black, and finishing the piano's white keys with gloss enamel spray paint. Grey keys are used for non-musical input, white and black to mimic the piano. Unpainted, the black keys retain their original labels, ensuring that visual cues for the less-familiar top row of symbols and punctuation remain. However, the gloss finish of the white keys obscures labels for two rows of letters. In a consumer version, it would likely be necessary to include labels, though the target user is assumed to have motor skill and knowledge of basic keyboard layout, enabling a stronger piano aesthetic. The finished prototype is pictured in Figures 1 and 7.

With completion of the prototype hardware, the final stage of development concerned calibrating pressure readings, to account for the location of the sensor and its relative proximity to key presses (Figure 8). For keys farther from the sensor, sensitivity is reduced, requiring an additional scaling factor to normalize the device's response to touch across the range of pitches. For such calibration (and in the absence of a precise method to reproduce exact key pressures), sample data was collected through a series of structured manual input sequences (for example, descending musical turns), wherein multiple readings from different postures, hands, fingering, and users were averaged to build a generic model of the changing sensitivity across the keyboard. From this, a map of reciprocal values was constructed to compensate for the variation in sensitivity.

4. USER EVALUATION

To evaluate interaction and expression using the device, a controlled ecosystem was developed to support a user study, combining the prototype hardware with a text-based software pattern sequencer, designed to provide a minimal music editing UI for testing touch, expressivity, and performance in mixed-mode (music and computer) input (see video). Users spent 30-minutes with the device, divided into four stages: two free play / practice stages to acclimatize subjects to the technology, respectively with and without velocity-sensitivity; then two stages of more purposeful composition, creating a short excerpt of music, similarly with and without velocity-sensitivity. The software provided a selection of velocity-sensitive, sample-based voices, including piano, EP, tuned percussion, oboe, flute, cello, and drum kit. All data input was captured and stored for future analysis.

Upon completion, subjects completed a survey that quantitatively probed (using an 11-point Likert scale, scored 0 to 10 or -5 to +5) their subjective experience of various aspects of using the hardware, and relevant prior experience of music, computing, and keyboards. An additional comment section allowed subjects to qualitatively feedback on their experience. Participants (n=15) were drawn from the students and staff of UWE's music technology course, reflecting a variety of computer music practices and aesthetics. Despite the relatively small sample size, results (see Figures 9, 10) demonstrated good consistency, as discussed in the remainder of this section.

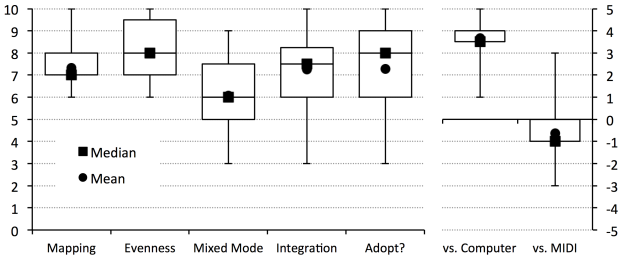


Figure 9. Box Plot of User Study Responses (n=15), including evaluations (left) and comparisons (right).

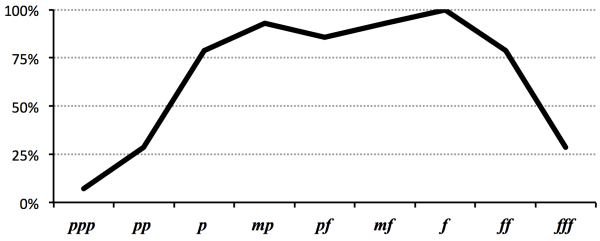


Figure 10. Dynamic Sensitivity Map (from user study responses, n=15).

On scales of 0 to 10, users reported very good precision ($\bar{x}$=7.33, σ_x=.98) in mapping between touch pressure and resulting velocity, as represented in both the visual notation and audio output. Several comments highlighted "surprising" and "unexpected" levels of responsiveness in the keyboard. Consistency (evenness) of touch across the keyboard similarly scored highly ($\bar{x}$=8.00, σ_x=1.46).

While the sensor is 10-bit (0-1024) and MIDI velocity / aftertouch encoding 7-bit (0-127), the exact precision – accounting for sensor and electrical noise – is unknown, but users were able to reliably capture a minimum of 6 practical dynamics levels (from ***p*** to ***ff***) using the keyboard. This was assessed using a question in which subjects circled dynamics marks (from ***ppp*** to ***fff***), corresponding to those they felt able to reproduce and distinguish, as shown in Figure 10.

The results, as well as a number of comments, indicate that very soft touches were difficult to capture. This is likely due to the physical key mechanism, where the silicon domes provide sufficient haptic resistance to create a minimum pressure 'threshold' for triggering a keystroke, combined with the mapping function used in the code. Through initial testing, a basic $y = \sqrt{x}$ mapping (Figure 11) was applied to raw sensor readings to open up the range of velocities produced, enabling more varied and expressive control. While the results show this non-linear mapping was broadly effective, lower dynamics map to a very limited range of very low pressures – notably, also at the extremes of the accelerometer's stated sensitivity. To compensate, an alternative mapping is also illustrated in the figure, integrating a linear section to expand the range yielded by lower pressures. Initial tests suggests this strategy is effective at extending response to softer dynamics, but further study is needed to assess how this impacts the overall feel or 'touch' of the keyboard.

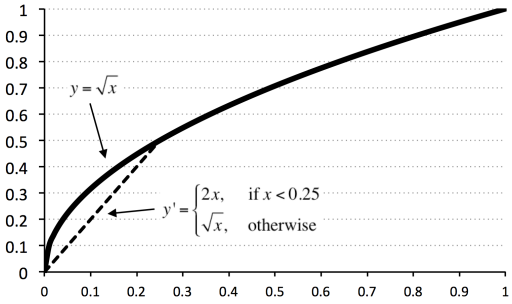


Figure 11. Control Mapping Functions, including original (*y*) and revised (*y'*).

Mixed-mode interaction scored reasonable high ($\bar{x}$=6.07), but with higher variance (σ_x=1.98). This category assesses how users are able to mix musical input (notes) with computer input (symbolic entry, cursoring, clipboard, playback, program control, etc.). As such, it depends on computer and keyboard experience, which varied within the sample. However, comments from more experienced computer musicians and keyboard users noted benefits when composing via a single integrated device – that it "focused" or "accelerated" their interaction. Users also appreciated the facility to easily and quickly "fine-tune" MIDI-recorded velocities using alphanumeric entry.

The velocity-sensitive functionality of the device adds significant musical expressivity to the standard computer keyboard. On a scale of -5 to +5, comparisons were consistently and overwhelmingly positive ($\bar{x}$=+3.67, σ_x=1.18). Comparisons with specialist music input devices (MIDI keyboards, controllers, etc.) were less favorable, though by a narrower margin than expected ($\bar{x}$=-.64, σ_x=1.60). Comments suggest that while the touch and real-time performance characteristics of the keyboard are inferior, many users recognize a form of non-realtime, meta-expressive power that arises from the symbolic and more abstract music editing functionality enabled through integration with computer functionality (program control, clipboard use, *ad hoc* playback) – an "enjoyable", high-energy, immersive, and focused way of writing music; rapid and fluid, but decoupled from 'live' musical time (c.f. [10]).

When asked how easy it might be to integrate the device into their existing computer music practice, subjects responded positively ($\bar{x}$=7.25, σ_x=1.86). This also translated to a similarly high average likelihood that subjects would adopt such a device, if available at a suitable price point ($30-60) ($\bar{x}$=7.29, σ_x=2.27). While this project is not intended as a commercial enterprise, such questions can reveal factors in the design of new interfaces for musical expression that facilitate (or inhibit) wider adoption. In this instance, the simple design adds an affordable and practical level of expressivity to a generic computer input device, without compromising compatibility with existing uses. For desktops, it replaces the existing keyboard, requiring no additional space, and uses driverless USB connectivity for 'universal' compatibility. While less complementary to laptop setups, survey comments also highlight potential for "portable" use, similar to existing small form-factor MIDI controllers.

Finally, a number of comments (as well as verbal feedback) from participants simply noted how much more "enjoyable" or "fun" the interaction style was, compared to conventional computer music methods, such as mouse and keyboard manipulation of sequencers and DAWS. This correlates with a large number of subjects continuing to use the system beyond the allotted 30-minute window of the experiment, and expressions of interest in future development of the technology.

5. DISCUSSION & FUTURE WORK

The velocity-sensitive computer keyboard described here, especially in light of user feedback, highlights opportunities for expressive extensions to ubiquitous computer input devices. The development of more expressive functionality, benefiting artistic creativity but using a device optimized for productivity, demonstrates the potential for new interfaces for expression in the non-realtime and "offline" modes of notation-mediated interaction that characterize computer work. To that end, this paper has explored applications in music editing software, such as sequencers, trackers, DAWs, and score editors; focusing on expressive interfaces for composition (plus arrangement and transcription), rather than live performance – augmented computer device, rather than augmented instrument. However, one area of live computer music that may benefit from an integrated device for both expressive and symbolic control is live coding: whether the device presented here would help avoid mode switching between code and controller, or whether there are more novel applications for pressure-sensitivity (in code editing itself) is an interesting direction for future work. Moreover, basic pressure and velocity sensitivity have applications in many areas of computing and digital creativity, such as games or expressive text writing [1][2].

As concerns the device discussed here, testing and user feedback identify several directions for continued development. Further improvements in sensitivity, response, and mapping can be affected through minor refinements of the embedded code (envelope following, filtering, and other DSP). Accurate testing could improve calibration, using specialist test equipment to reproduce precise pressures (available through the university's product design department). Other aspects of 'touch' relate to physical characteristics, and alternatives to the basic membrane / silicon dome key mechanism can be found in other keyboard designs that offer softer, quieter keys with a more linear haptic resistance (e.g. Cherry MX Black mechanical switches, and other 'quiet key' keyboards). As previously observed, improved sensor chips (or combinations) may also offer more sensitivity.

Using a three-axis accelerometer allows expressivity in two further dimensions, which can be explored through the existing prototype hardware. However, while the impact of hitting a key produced practical readings and resolution (Z-axis), initial tests of X- and Y-axes showed only minimal fluctuations when a pressed key was pulled or pushed left/right or up/down. Other envelope following and signal processing techniques may improve the utility of these small signals, to add modulation of other musical properties – and potentially fingering methods such as those exposed by the *SeaBoard* [6] or *TouchKeys* [7] interfaces. At the same time, an alternative interaction style is possible, where the user triggers a note via a key with their left hand and manipulates the entire keyboard with their right, possibly using the cursor cluster (immediately adjacent to the sensor) as a grip for lateral and vertical movement. In this way, the keyboard becomes a form of 2D pitch bend / modulation wheel. Indeed, this bimanual interaction style fits with postures observed in computer music interaction (Figure 2), where data manipulation using alphanumeric keys and navigation using cursors or mouse are respectively split between the hands.

The low-cost accelerometer approach is easily adapted to other keyboard layouts, while alternative form-factors might also be considered. For performing artists, the laptop is a popular platform. Laptops with mechanical drives typically feature an accelerometer (e.g. Apple's *Sudden Motion Sensor*) to detect shocks, used to protect the sensitive hardware. This would enable an elegant software solution, adding velocity readings to laptop-based music software without any additional hardware. However, with the advent of solid-state drives, such sensors are now rare, though an attachable USB accelerometer could be used; albeit requiring a software layer to integrate keystrokes and pressure detection, such as that used in the prototype. Moreover, the presence of accelerometers in most mobile devices would enable a similar extension of expressivity for input to touch-based devices, such as tablets and smartphones, which are increasingly finding new uses in music making and performance.

Finally, for the next stage of this project; a more challenging technical milestone will be to realize the integrated hardware device, using the architecture in Figure 3 (a): offering combined USB HID and USB MIDI functionality, fully housed within the keyboard chassis, connected by a single USB cable, and supporting driverless operation without dependence on client software support. As with the prototype, existing technologies and cheap components are available to facilitate such development, but the manufacturing process becomes significantly more complex and expensive, moving from prototyping boards to custom PCBs and hardware components. The currently proposed design combines a Microchip PIC24/PIC32 (16-bit MCU with USB MIDI support and 10-bit ADCs) and USB2512B (2-Port USB Hub IC). Additional funding (possibly crowd-sourced) is being sought to continue the project.

One of the principle objectives of this research is to explore unified interfaces for computer music that help maintain focus and workflow in software-based composition and production processes, integrating direct, low-level musical input (live performance) with more abstract, high-level computer-based editing (shortcuts, clipboard, ad-hoc playback, arranging, etc.). Identifying and addressing usability issues in existing packages (i.e. sequencers and DAWs) such as focus, device, and context-switching, delayed or deferred edit feedback, and UI or notation inconsistencies has been a focus of previous research on flow in computer music [8-11], which it is hoped the development of hardware input devices might inform. To this end, a larger scale deployment and extended, longitudinal study of an integrated, end user-ready edition of the device – following different users, applications, environments, and workflows over an extended period – represents a promising direction for future study.

6. SUPPORTING VIDEO

A video presenting expressivity tests and demonstrating mixed-mode interaction with the device, within a computer music scenario, is available from: http://revisit.info/nime2016.

7. ACKNOWLEDGEMENTS

This research was funded by UWE's Department of Computer Science and Creative Technology. Many thanks also for the enthusiastic participation and ongoing feedback of students and staff who took part in the user study.

8. REFERENCES

[1] Buxton, W. *Multi-touch systems that I have known and loved.* Microsoft Research. 2014. Available at: http://www.billbuxton.com/multitouchOverview.html.

[2] P.H. Dietz, B. Eidelson, J. Westhues, and S. Bathiche. A Practical Pressure Sensitive Computer Keyboard, *In Proceedings of UIST'09*, 2009, 55-58.

[3] D C. Dobrian and D. Koppelman. The 'E' in NIME: Musical Expression with New Computer Interfaces. *In Proceedings of NIME'06* (Paris, France, June 4-8), 2006.

[4] A. Feit and A. Oulasvirta. PianoText: Redesigning the Piano Keyboard for Text Entry. *In Proc. of the ACM conference on Designing Interactive Systems 2014 (DIS'14)* (Vancouver, Canada, June 21-25, 2014), ACM Press, New York, NY, 2014.

[5] P.N. Juslin, A. Friberg, E.Schoonderwaldt, and J. Karlsson. Feedback Learning of Musical Expressivity. *In Musical Excellence* (ed. A. Williamon), OUP, 2004, 247-270.

[6] R. Lamb and A. Robertson. Seaboard: a new piano keyboard-related interface combining discrete and continuous control. *In Proc. of NIME'11* (Oslo, Norway, May 30 - June 1), 2011.

[7] A. McPherson. TouchKeys: capacitive multi-touch sensing on a physical keyboard. *In Proceedings of NIME 2012* (University of Michigan, Ann Arbor, MI, US, May 21-23.

[8] C. Nash. Supporting Virtuosity and Flow in Computer Music, *PhD Thesis*, University of Cambridge, 2011.

[9] C. Nash and A. Blackwell. Tracking Virtuosity and Flow in Computer Music. *Proceedings of ICMC 2011* (Uni. of Huddersfield, UK, July 21-August 5), 2011, 575–82.

[10] C. Nash and A. Blackwell. Liveness and Flow in Notation Use. *In Proceedings of NIME 2012* (University of Michigan, Ann Arbor, MI, US, May 21-23), 2012, 28–33.

[11] C. Nash and A. Blackwell. Flow of Creative Interaction with Digital Notations. *In Oxford Handbook of Interactive Audio* (eds. K. Collins, B. Kapralos, and H. Tessler), Oxford University Press, NY. 2014, 387-404.

[12] M. Smyth, A. Collins, P. Morris, and P. Levy. *Cognition in Action* (2nd Edition). Lawrence Erlbaum Associates, 1994.