Practical Considerations for MIDI over Bluetooth Low Energy as a Wireless Interface

Johnty Wang
Input Devices and Music Interaction Lab/CIRMMT
McGill University, Canada
johnty.wang@mail.mcgill.ca

Axel Mulder
Infusion Systems
Montreal, Canada
axel@infusionsystems.com

Marcelo M. Wanderley
Input Devices and Music Interaction Lab/CIRMMT
McGill University, Canada
marcelo.wanderley@mcgill.ca

ABSTRACT

This paper documents the key issues of performance and compatibility working with Musical Instrument Digital Interface (MIDI) over Bluetooth Low Energy (BLE) as a wireless interface for sensor or controller data and inter-module communication in the context of building interactive digital systems. An overview of BLE MIDI is presented along with a comparison of the protocol from the perspective of theoretical limits and interoperability, showing its widespread compatibility across platforms compared with other alternatives. Then we perform three complementary tests on BLE MIDI and alternative interfaces using prototype and commercial devices, showing that BLE MIDI has comparable performance with the tested WiFi implementations, with end-to-end (sensor input to audio output) latencies of under 10ms under certain conditions. Overall, BLE MIDI is a viable solution for controllers and sensor interfaces that are designed to work on a wide variety of platforms.

Author Keywords

MIDI, Sensor Interfaces, Latency

CCS Concepts

•**Hardware** → **Sensor devices and platforms;** •**Applied computing** → *Sound and music computing; Performing arts;*

1. INTRODUCTION

Digital Musical Instruments [12] constitute a subset within the general category of interactive digital systems. In such systems, a key part of the process involves transmission and reception of control data arising from sensor interfaces [13] or intermediary processing modules. Performance characteristics and compatibility are main issues when implementing such interfaces, and in this work we describe an interface that implements the messaging specifications of Music Instrument Digital Interface (MIDI) protocol over a Bluetooth Low Energy (BLE) connection, henceforth referred to as BLE MIDI.

In this paper, we first present examples of available wireless hardware and communication protocols, which are combined to represent alternative interfaces to BLE MIDI. Then, we present an overview of the BLE MIDI standard along with some theoretical performance characteristics. Finally, we describe the implementation and testing of various custom built and commercially available BLE interfaces with other existing protocols and discuss the benefits and limitations of using BLE MIDI.

NIME'19, June 3-6, 2019, Porto Alegre, Brazil

2. BACKGROUND AND RELATED WORK

There are two main aspects of BLE MIDI that should be mentioned with regards to related systems. First is the hardware specification for the interconnection between two systems, and second is the protocol for supporting the interchange of data. Any alternative to BLE MIDI subsequently described and compared with this work are first contextualized using both the hardware features as well as software protocol, since these aspects work in tandem to influence performance parameters and compatibility. Combining a specific hardware specification with a protocol yields a particular interface for comparison.

2.1 Wireless Connections

One of the obvious features of Bluetooth as an *interface* is its wireless nature. The freedom of an untethered device may be attractive for many applications, and the reliability of wireless connections have greatly improved over the years [3]. Nearly all mobile computational devices on the market today contain hardware that supports IEEE 804.15.4 (Bluetooth) and 804.11a/b/g/n/ac (WiFi). We acknowledge the use of other wireless radio systems such as ZigBee, as applied in the case of Sense/Stage interface system that has been successfully used in various artistic contexts [1]. Similar systems such as the nRF24L01+, like ZigBee, possess similar features and are especially interesting due to their mesh networking capabilities that are useful when scaling up to large number (100+) devices [5]. However, both these systems require additional hardware not found on commodity computational devices, and while an adapter dongle with custom driver software can be easily added to a desktop computing environment, the same cannot be said of mobile platforms. WiFi is supported on virtually all mobile devices currently in operation, while BLE MIDI is supported by all Android and iOS devices from three or four generations ago up to the date of this article. Therefore, WiFi and Bluetooth are significantly easier to work with where hardware compatibility (especially with mobile devices) is needed.

2.2 Protocols

The MIDI messaging protocol, due to its widespread manufacturer support, became extremely popular shortly after its release [8]. Most modern music synthesis and production software today, as well as most applications dealing with digital media (such as visual and show control systems) will

Table 1: Additional dependencies for implementation, H = hardware; S = software

Interface	**iOS**	**Android**	**macOS**	**Windows**	**Linux**
MIDI(BLE)	None	None	None	None*	None
RTP-MIDI (WiFi)	None	S	None	S	S
MIDI(USB)	H	H	H	H	H
OSC(WiFi)	S	S	S	S	S
Other(Wireless)	H,S	H,S	H,S	H,S	H,S

*See Section 2.7 for details on Windows OS

support MIDI. Specifically, the General MIDI Standards[1] define particular sets of messages that, while limiting (7 or 14 bit integers for parameter values), will ensure compatibility between devices. For custom data types, it is possible to pack arbitrary byte streams via System Exclusive (SysEx) messages, which can be parsed by a supporting receiver.

More recent standards such as OpenSoundControl (OSC) [17] provide far more flexibility by providing a user defined messaging namespace and custom message structures. By utilizing the User Datagram Protocol (UDP)[2] that sits on top of the network stack, messages can be formatted using human-readable plain text without significant impact on performance due to the high bandwidth available. Strings, arrays of values, and floating point numbers in addition to raw bytes (for compactness) can be transmitted via OSC. However, the main consequent trade-off for this flexibility is that both sender and receiver applications must be configurable to parse the messages. Additionally, any device that employs OSC must implement the networking stack and transport mechanisms required. While this requires additional processing compared with MIDI, this requirement is becoming increasingly trivial with modern embedded systems since many microcontrollers provide the necessary processing power and hardware interfaces. However, there are much fewer end-user applications that support OSC versus MIDI.

2.3 Interfaces and Interoperability

Combining the hardware specification and data exchange protocol yields the concept of an *interface* that provides a physical realization of the connection.

2.4 OSC over WiFi

The WiSe Box [4] and x-OSC [9] are two examples of WiFi based general purpose sensor acquisition systems that employ OSC for transmitting sensor data to a host computer. The former did not report specific results, while the latter exhibited measured latencies of around 10ms and throughput of 400kbps. Since most modern devices contain WiFi capabilities, this *interface* is an ideal choice where the receiving software is capable of receiving and parsing OSC messages.

2.5 RTPMidi over WiFi

Real Time Protocol MIDI (RTP-MIDI) provides a mechanism to transmit standard MIDI messages over a network, and can be theoretically implemented on any hardware system that supports OSC since it uses the same UDP networking stack. With the addition of a software driver bridge, it is possible to implement a network MIDI port on a receiving device. The physical performance of such a link would be similar to OSC, but with a simpler messaging protocol that has its associated strengths and drawbacks. Commercial adapters that convert MIDI to RTP-MIDI exist, but the receiving side must implement software support to create a virtual RTP-MIDI compatible port.

2.6 MIDI over BLE

BLE MIDI transfers standard MIDI data over a BLE connection. Making use of the ubiquitous nature of BLE hardware on modern devices and the availability of native driver support under all desktop and mobile operating systems, once a BLE MIDI device is paired, the data is directly available by most receiving software. The details of the implementation is described in more detail in Section 3.1.

2.7 Compatibility Comparison

Table 1 presents a compatibility chart showing devices running various operating systems and the hardware/software dependencies to support each *interface*. This table applies to *currently* available devices on the market that almost universally contain WiFi and Bluetooth 4.0-capable radios, which are features that can be easily added to a non-wireless desktop computer. Wired MIDI, in the form of a class-compliant USB-MIDI interface adapter is presented for comparison and is universally supported in software by all devices via a regular USB port or adapter dongle in the case of mobile devices.

As the compatibility table shows, BLE MIDI is the best supported system combination from both a hardware and software perspective compared with other alternatives. The only operating system environment where an additional bridging application is required is Windows 10 since a BLE MIDI port is a separate entity inoperable with "standard" MIDI ports. On all other systems, a connected BLE MIDI device will simply appear as a native MIDI port, so any existing MIDI application will be able to communicate with a BLE MIDI device.

3. MIDI OVER BLE

3.1 Implementation

The BLE MIDI standard is defined using a Bluetooth Generic Attributes (GATT) profile as part of the BLE standard[3]. The profile defines how the device must advertise its presence, initiate connection with another device, and handle the transfer of information once the connection is established.

There are a number of hardware-specific commercial and open source Bluetooth development environments available that can be used to build a BLE MIDI interface, with some popular integrated microcontroller solutions including Nordic Semiconductor's nRF series[4] and Espressif Systems'

[1] `https://www.midi.org/specifications/category/gm-specifications`

[2] OSC can be transported via other means, but this is the most common

[3] `https://www.bluetooth.com/specifications/gatt`

[4] `https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy`

ESP32[5]. The Arduino IDE has open source libraries for a number of BLE capable platforms.

Based on the specification, a few key performance metrics can already be determined, and appear in the following sections.

3.2 Theoretical Latency

Wired MIDI operates at a baud rate of 31250bps, and each message byte is transmitted using 10 bits [7]. Therefore, the minimum time it takes to send a 3-byte (30 bit) message over the wire is 0.96ms. In practice, the lowest measured latency for MIDI signals are slightly longer, especially when going through interfacing hardware on a computer and have been measured to be on the order of a few milliseconds, with USB based systems that also transmit audio data perform slightly worse [16]. This is not surprising since the bus must handle both MIDI and audio data at the same time. At the same time, modern USB-based MIDI interfaces are known to provide the lowest measured latency in a comparison of interfaces [11].

Since Bluetooth is a packet based protocol [2] and transmission can only occur within connection interval events, this interval will determine the lowest possible period between successive samples. Based on the standard, the minimum connection interval is 7.5ms for BLE, which is theoretically within the 10ms limit required for musical applications [15] but may not leave much room for the rest of the signal processing and synthesis chain. The BLE MIDI spec requires the interval to be 15ms[6]. Later in practical measurements we see the connection interval, as tested on a modern desktop operating system, is in fact higher at 11.25ms. The consequence of this is that if incoming data cannot be transmitted within the current connection interval, it must wait until the next one with a relatively large gap between transmissions.

One redeeming factor of BLE MIDI is the addition of a millisecond timestamp to each message, which allows the exact signal acquisition time to be transmitted with a MIDI message sent over the BLE link. This means that despite the longer latency, it is possible to preserve the exact timing characteristics of the signal, which may be useful for applications where timing accuracy is more critical than low latency.

3.3 Theoretical Bandwidth

The overall bandwidth of Bluetooth 4.0 is 1 MBps. However, this is the maximum capacity of the physical channel and must include the lower level support structures of the networking stack. The theoretical maximum depends on a number of variables, including the number of packets that can be sent within a single connection interval which varies by host operating systems, and varies between 56kbps and 128kbps depending on a number of parameters enforced by the receiving operating system [6]. Based on these values the actual bandwidth of BLE is actually much closer to the MIDI standard of 31.25kbps as carried via the 5-pin serial connection[7], and far from the rates of modern USB and networking protocols that are capable of passing data on the order of hundreds of MBps. This is perhaps not surprising as BLE was intended for low power signal acquisition purposes rather than high throughput. The obvious consequence of BLE's low channel capacity means that it is unsuitable for delivering large amounts of data.

[5] `https://www.espressif.com/en/products/hardware/esp32/overview`

[6] `https://www.midi.org/specifications-old/item/bluetooth-le-midi`

4. PERFORMANCE COMPARISONS

In this section we describe empirical performance comparisons between BLE MIDI and other wired and wireless interfaces. We make use of devices and platforms that are widely available to the NIME community including cheap microcontroller systems, commodity computing devices and software. We present the configuration and results from three main tests: First we performed end to end latency tests. Then we evaluated the roundtrip latency of a commercially available adapter that added BLE MIDI capability to an existing wired MIDI port. Finally, we compare the available bandwidth of the BLE MIDI connection with alternatives. While these tests are significantly different in nature, they present cases where the wireless BLE link can be isolated and compared directly with another alternative.

4.1 End-to-End Latency Test

During the design of a general purpose sensor interface, we had an unique opportunity to isolate the BLE link from an existing wired connection while keeping the rest of the hardware components identical. Figure 1 shows the previous and newly developed sensor interfaces. The new interface provides both a wired USB connection as well as a BLE connection (as a *peripheral* device) and can operate in either mode. This provides a configuration that can test any differences in performance after introducing the BLE MIDI to the processing chain.

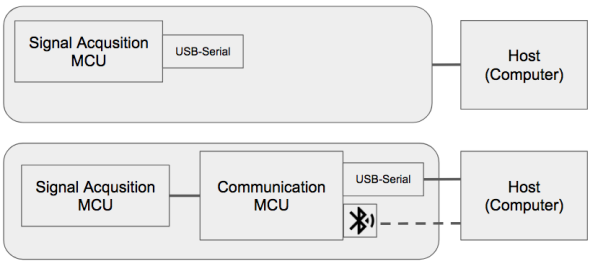

Figure 1: **Top**-Existing Sensor Interface with USB Serial; **bottom**-New interface with USB Serial and BLE MIDI

We constructed the test rig as described in a previous study testing end to end latency [11] that measures the time between a sensor input trigger and the final synthesized audio output on a receiving host, and employ the lowest latency Max/MSP software configuration (overdrive on, 32-bit vector size). While the results from this test embeds a number of additional processing steps that contributes to the overall latency, we are able to fix the other variables to provide a comparison between just the wired and BLE interfaces. To first make sure our general test configuration is operating correctly, we also constructed an identical USB-MIDI based Teensy system from the previous work, and it reproduced very similar results. We also implemented a minimal system that contained a single digital input trigger that transmitted either a BLE MIDI, RTP-MIDI (over WiFi), or OSC (over WiFi) message and compared the results. Figure 2 shows the test configuration for the custom interface (USB and BLE MIDI), and the minimal test case between BLE and WiFi connections.

As shown in Table 2, we first see that the Teensy implementation yielded very similar results to the previous study [11], and the marginally better performance we recorded may be attributed to a slightly faster computer. Our custom interface using the USB-Serial connection had performance similar to the Teensy, but when the BLE MIDI link

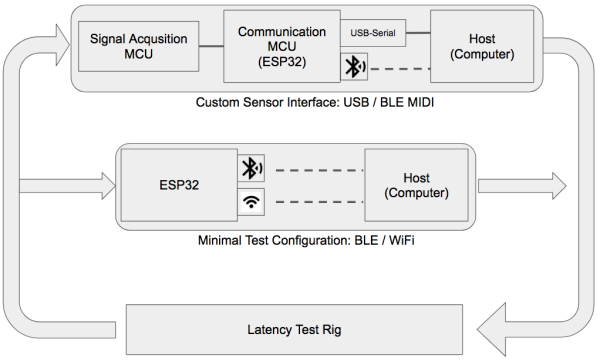

Figure 2: Test setup for the custom sensor interface and minimal examples

Table 2: End-to-End Latency Measurements

Connection	Latency	Std Dev
USB MIDI (Teensy)	4.1 ms	0.4 ms
USB MIDI (Teensy)*	5.1 ms	0.4 ms
USB-Serial (custom)	6.2 ms	0.35 ms
BLE MIDI (custom)	19.1 ms	2.7 ms
BLE MIDI (minimal)	7.5 ms	1.8 ms
WiFi RTP-MIDI (minimal)	8.5 ms	8.0 ms
WiFi OSC (minimal)	7.6 ms	2.9 ms
WiFi OSC*	6.7 ms	1.5 ms
BT 2.0-Serial	30 ms	14.6 ms
BLE*	139 ms	21.9 ms

* indicates results from previous study [11]

was used instead, latency was significantly higher. Looking at the minimal configuration on the ESP32 microcontroller running BLE MIDI, WiFi OSC and WiFi RTP-MIDI, we see relatively comparable behaviour between the wireless configurations. The custom BLE interface exhibited about 11ms more latency than the minimal case, possibly due to internal sensor processing which resulted in the transmission of the message in a subsequent BLE connection interval.

Compared with other Bluetooth based implementations, we see that the BT 2.0-Serial interface, employed by a previous version of our sensor interface, exhibited significantly higher latency than BLE. The final, extremely large latency value in the table that reports measured latency from a BLE interface [11], as explained by the original authors, could be attributed to the polling method since the BLE MIDI implementation was not used.

4.2 Commercial BLE MIDI Adapters

There are a number of commercially available BLE MIDI controllers that operate as BLE MIDI peripheral devices. A non-exhaustive list appears in Appendix A. Two interesting adapter devices, the Yamaha MD-BT01 and CME Widi Bud, allow BLE MIDI capability to be added to existing systems. The Yamaha device is a "BLE bridge" adapter that allows a wired MIDI port to send and receive BLE MIDI messages. Conveniently powered by the small amount of power available on a MIDI Output port, the Yamaha adapter provides an easy way to turn any MIDI device into a BLE MIDI peripheral and communicate with a host receiving application. The CME device allows any host device with a USB port to operate with other BLE MIDI peripherals.

To test the implication of retrofitting an existing device with a BLE MIDI connection, we created a test configuration consisting of concurrent wired USB and BLE MIDI connections between two computers. An M-Audio MIDISport 2x2 was used on Computer 1 (2010 Mac Pro Tower, Quad-core Xeon 2.8G hz) with A connected to a Roland UM-One 1x1 on Computer 2 (2014 Macbook Pro, 2.5 Ghz i5). The Yamaha MD-BT01 was connected to Port B of the MIDISport and a BLE MIDI link established with the built in BLE interface of Computer 2. By measuring the timing of messages sent through these two links, we can observe the effect of replacing the wired USB MIDI port on the second computer with a BLE link, as shown in Figure 3.

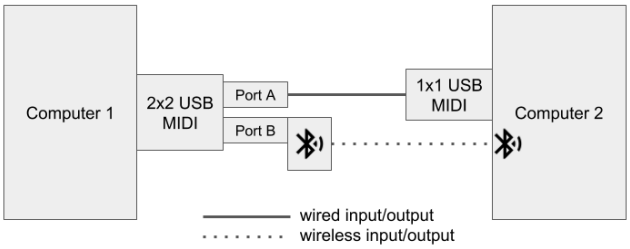

Figure 3: Wired and BLE MIDI Test Configuration

A simple MIDI timing application was written in C using the RtMidi library in MacOS [14]. The application opens a MIDI input and output port, and performs one of two roles:

- **Sender**: Emits a note-on message and measures the interval until a message is received and outputs to screen. Performs this operation in a loop with a preset delay between the next test.
- **Receiver**: Emits a note-on message whenever a message is received.

A flow diagram of the application is presented in Figure 4 showing the operation of the sender and receiver modes.

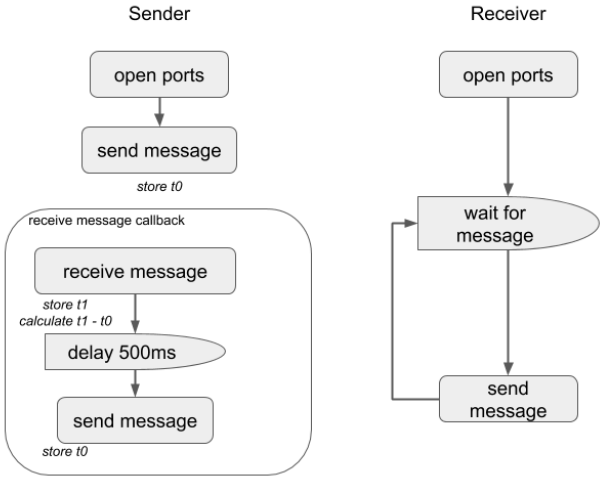

Figure 4: Flow Diagram of MIDI roundtrip latency measurement

When the sender and receiver's MIDI input and output ports are connected to each other, the sender will effectively measure the round-trip delay of the MIDI channel. We took measurements of 1000 samples with an inter-message delay of 500ms, and the results are presented in Table 3, with Computers 1 and 2 taking turns being sender and receiver, respectively. The results show that the addition of the BLE

MIDI link resulted in nearly 26ms of additional delay for the roundtrip, or about 12ms for one way. Similar to the end-to-end latency measurements between the USB and BLE interfaces of our custom interface in Section 4.1, this added delay may be accounted for by the additional BLE connection intervals.

Table 3: Round-trip Delay of wired and BLE interface

Connection	Round-trip	Std Dev
Wired Computer 1 to 2	4.3 ms	2.2 ms
BLE Computer 1 to 2	29.9 ms	15.0 ms

4.3 Bandwidth Tests and Packet Inspection

Since the saturation point of a wireless channel depends on many changing environmental factors, we attempted to perform a rudimentary test using the BLE devices available on hand to obtain a general idea of the overall bandwidth of a BLE MIDI connection, and compare that with WiFi links where possible. A test BLE application was implemented on the ESP32 and nRF51822 microcontrollers that emits increasing amounts of synthetic data via MIDI System Exclusive messages. Using Apple's BLE PacketLogger application on the receiving computer, we found that both radios, when using the BLE MIDI interface, were able to transmit up to around 40kbps before packets were no longer being received at the nominal transmission intervals. This is significantly lower than the theoretical values described in Section 3.3. Running a similar test on the ESP32 operating as an WiFi OSC sender instead, the total bandwidth was nearly double. Other WiFi radios such as the x-OSC interface [9] perform at much higher rates.

One other interesting outcome obtained through the analysis of the packets received by MacOS is that the connection interval chosen by the OS appears to be 11.25ms. While this conforms to the requirement of being less than 15ms, this value is larger than the minimum possible connection interval of 7.5ms. This suggests that access to and modification (if possible) of the operating system Bluetooth driver could potentially yield better performance. Another consequence of the connection interval is that internal processing delays beyond a certain threshold would result in the addition of an entire connection interval to the latency.

5. DISCUSSION

In this section we first describe some of the implications of the results obtained in the evaluation, and list some limitations of the evaluation and ways in which it can be extended.

5.1 Suitability of BLE MIDI

Given its significant advantages of compatibility, BLE MIDI is an ideal choice for general purposes sensor interfaces and controllers. As shown in Table 1 in Section 2.3 we see that BLE has nearly universal support on almost every platform, which is better than any other available alternative. One example where this could be useful is when building a controller interface that can work with a wide variety of existing software applications. A custom sensor interface or controller that contains built in mapping to MIDI messages could be used by a large number of MIDI-compatible sequencing and synthesis software running on desktop computers, laptops, and tablets out of the box if it supports BLE MIDI. This can serve to explain the choice of the increasing number of commercial MIDI controllers that have this capability (a non-exhaustive list presented in Appendix A), since in this application, the controller is designed to work with a potentially large number of receiving devices/software running on different platforms. The minimally measured latency of a BLE interface is comparable with WiFi, and while the bandwidth appears to be worse, is comparable with the original MIDI specifications. It is possible to make existing MIDI-compatible devices wireless using commercially available BLE MIDI adapters, at the cost of significantly higher latency.

Due to some of the advantages of BLE MIDI, we have, in collaboration with *Infusion Systems*, recently released the *I-CubeX WiDig,* a new sensor interface that adds BLE MIDI functionality (in addition to USB and potential for WiFi). Since the sensor interface contains configurable on-board mapping to MIDI messages from sensor data, the added BLE MIDI capability allows a custom sensor setup to seamlessly connect to a large number of applications (such as GarageBand on iOS), and synthesizers running on desktop platforms. We are also experimenting with a new version of the *T-Stick* [10], a controller developed over the past 10 years for research in digital musical instruments, with built-in mappings that generate BLE MIDI output to explore the potential conveniences that the interface affords.

On the other hand, if one is building a system from scratch or is able to provide additional software integration and/or tools that already support other protocols like OSC, WiFi devices can be used for better bandwidth. When using networked devices, RTP MIDI is another viable alternative that retains some of the compatibility benefits of standard MIDI while employing a higher performance wireless channel. Finally, if both ends do not need to conform to the limited speed of the "classic" serial connection via the 5-pin DIN connection, USB MIDI can operate at much higher rates (up to full USB speed[7]) over a wired connection.

5.2 Scope of Evaluation

The evaluations performed and described above may be improved in many ways. For example, the latency of these connections are often a function of other parameters such as packet size and sampling interval since the saturation of available bandwidth will affect the response time [9], resulting in a more comprehensive test that integrates the latency test described in Section 4.1 with the bandwidth test in Section 4.3. Additionally, the number of concurrent devices may be another parameter of interest for certain applications.

One other significant factor that was not tested was general reliability metrics like connection stability, and perhaps more importantly, the range of the connection. Therefore, the conclusions made in this evaluation can be supplemented with further tests to obtain these parameters, ideally under a wide variety of conditions including performance settings with other equipment (such as stage lighting, wireless audio systems, etc).

Power consumption, and its subsequent effect on battery life and shape/size/weight parameters is yet another consideration for portable wireless applications, but once again beyond the scope of this evaluation. It should be noted however that modern battery power densities typically mean a suitable pack can be found to meet the needs of most applications [3].

Finally, so far we have only compared and discussed the compatibility part of the cross platform capabilities. The performance analysis could be done on multiple platforms to see if the results are consistent across different operating systems using the same test procedures.

[7] `https://forum.pjrc.com/threads/46111-Teensy-USB-MIDI-Bandwidth`

5.3 Value of Standardized Tests

Finally, we acknowledge the importance of standardized testing procedures for these devices, and appreciate the efforts of previous research that attempted to increase the reproducibility of evaluations through the sharing of testing tools [11]. We follow this initiative by adding the new tools implemented and additional documentation such as the schematics of the hardware testing rig to a publicly available repository[8]. We hope that this will further increase the accessibility of these test procedures, to the benefit of the wider community.

6. CONCLUSION

In this paper we have described the implementation, use, and testing of MIDI via Bluetooth Low Energy as an interface for communication between modules in the context of digital interactive systems. Based on the features and measured performance characteristics, BLE MIDI is a potentially interesting replacement for wired MIDI interfaces due to its wireless capability and extensive hardware and software support in modern systems. A BLE MIDI device will transparently operate with a MIDI compatible application on most mobile and desktop platforms with no additions, and provides "out of the box" support for the most number of use cases compared with any wired or wireless alternative. The existence of commercially available BLE adapters allow both *peripheral* and *central* BLE MIDI devices to be added to existing wired MIDI systems or computers without BLE radios, at the cost of increased latency. BLE MIDI is a viable solution for any sensor interface that requires extensive interoperability with existing MIDI-compatible applications, and may serve to explain the growing number of consumer oriented musical interfaces that use BLE. At the same time, more extensive testing of other critical parameters such as range and general reliability are need before one can make a confident claim for using BLE MIDI in more serious applications.

7. ACKNOWLEDGMENTS

The primary author's research is funded through an NSERC Industrial Innovation Scholarship in collaboration with Infusion Systems

8. REFERENCES

[1] M. A. Baalman, V. De Belleval, J. Malloch, J. Thibodeau, C. Salter, and M. M. Wanderley. Sense/stage-low cost, open source wireless sensor infrastructure for live performance and interactive, real-time environments. In *Proceedings of the International Computing Music Conference*, 2010.

[2] Bluetooth SIG. Bluetooth core specification version 4.0. *Specification of the Bluetooth System*, 2010.

[3] P. R. Cook. Re-Designing Principles for Computer Music Controllers: a Case Study of SqueezeVox Maggie. In *Proceedings of the 2009 Conference on New interfaces for musical expression*, pages 218–221, 2009.

[4] E. Fléty. The Wise Box: a multi-performer wireless sensor interface using WiFi and OSC. In *Proceedings of the 2005 Conference on New interfaces for musical expression*, pages 266–267, 2005.

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

[6] K. Karami. Maximizing BLE Throughput on iOS and Android, Apr 2016. `https://punchthrough.com/blog/posts/maximizing-ble-throughput-on-ios-and-android` (Accessed 5 Aug 2018).

[7] P. D. Lehrman and T. Tully. *MIDI for the Professional: The Essential References for the Serious MIDI User*. Amsco, 1993.

[8] G. Loy. Musicians make a standard: the MIDI phenomenon. *Computer Music Journal*, 9(4):8–26, 1985.

[9] S. Madgwick and T. J. Mitchell. x-OSC: A versatile wireless I/O device for creative/music applications. In *Proceedings of the 2013 Sound and Music Computing Conference*, 2013.

[10] J. Malloch and M. M. Wanderley. The T-Stick : From Musical Interface to Musical Instrument. In *Proceedings of the 2007 Conference on New Interfaces for Musical Expression*, pages 66–69, 2007.

[11] A. P. McPherson, R. H. Jack, G. Moro, et al. Action-sound latency: Are our tools fast enough? In *Proceedings of the 2016 Conference on New interfaces for musical expression*. Griffith University, 2016.

[12] E. R. Miranda and M. M. Wanderley. *New digital musical instruments: control and interaction beyond the keyboard*, volume 21. AR Editions, Inc., 2006.

[13] A. Mulder. The I-Cube system: moving towards sensor technology for artists. In *Proceedings of the Sixth Symposium on Electronic Arts (ISEA 95)*, 1995.

[14] G. P. Scavone and P. R. Cook. Rtmidi, rtaudio, and a synthesis toolkit (stk) update. In *Proceedings of the 2005 International Computer Music Conference*, 2005.

[15] D. Wessel and M. Wright. Problems and prospects for intimate musical control of computers. *Computer music journal*, 26(3):11–22, 2002.

[16] J. L. Wright and E. Brandt. System-Level MIDI Performance Testing. In *Proceedings of the 2001 International Computing Music Conference*, 2001.

[17] M. Wright. Open sound control: an enabling technology for musical networking. *Organised Sound*, 10(3):193–200, 2005.

APPENDIX

A. EXAMPLES OF COMMERCIALLY AVAILABLE BLE MIDI DEVICES

Manufacturer	Model	Device Type
ACPAD	ACPAD	Guitar Controller
CME	X-Key Air	Keyboard Controller
CME	WIDI Bud	USB-BLE MIDI Adapter
Isla	KordBot	Chord Controller
Livid	Minim	Control Interface
Korg	micro/nanoKey	Keyboard Controller
Quicco	mi.1	MIDI to BLE MIDI
Roland	Aerophone AE-05	Wind Controller
ROLI	Blocks	Control interfaces
ROLI	Seaboard	Keyboard Controller
Sensel	Morph	Reconfigurable Interface
Yamaha	MD-BT01	MIDI to BLE MIDI
Yamaha	UD-BT01	USB-Host to BLE MIDI
Zivix	jamstik	Guitar Controller

[8] `https://github.com/idmil/bletests`