Analysis of Wireless Interface Latency and Usability for Digital Musical Instruments

Johnty Wang

Music Technology Area
Input Devices and Music Interaction Laboratory
McGill University
Montreal, Canada

November 2021

A document submitted to McGill University in partial fulfilment of the PhD Program in Music Technology.

2021/11/24

Abstract

The proliferation of mobile and wireless technologies has resulted in an increasing number of wireless interfaces and protocols that facilitate the transmission of data in the context of sensor interfaces and controllers for the implementation of Digital Musical Instruments (DMIs). Despite the large number of wireless devices and controllers available, there has been little systematic evaluation of these design choices from the perspective of DMI design. The diversity of choice creates potential confusion when selecting interfaces, made up of a combination of a physical interface (hardware), and messaging protocol (software). Specifically, the latency performance (of how responsive an instrument is, and how that might scale when additional devices are present in an ensemble setting), and usability (how easy it is to integrate with the user application) are often unspecified. This thesis presents answers to these questions by looking at two main issues: An empirical analysis of latency in wireless interfaces and the definition of tools to increase the usability of DMIs. A system for measuring end-to-end latency, where a sensor interface configuration is integrated into a system that responds to test trigger input and synthesizes audio output, is implemented to measure the latency performance of different wireless sensor interface configurations. Wi-Fi, Bluetooth, and LoRa are compared against wired USB/serial interfaces, with Open Sound Control (OSC) and MIDI as messaging protocols. Based on the findings of the work, Wi-Fi-based physical interfaces provide the most scalability in terms of number of concurrent devices, overall bandwidth, as well as minimal latency, although care must be taken when scaling beyond 5 concurrent devices operating within a single wireless channel. The interoperability of the interface will be dependent on the messaging protocol, and here there is a trade-off between flexibility of representation offered by a more open standard such as OSC on the one hand, and compatibility with existing standards such as MIDI on the other. The second part of this thesis then further explores the concept of interoperability and usability of sensor interfaces from the perspective of mapping frameworks, tools, and environments, including the recommendation for the incorporation of sensor interface characteristics as part of visual mapping utilities.

Résumé

La prolifération des technologies mobiles et sans fil a considérablement accru le nombre d'interfaces et de protocoles sans fil qui facilitent la transmission de données dans le cadre d'interfaces de capteurs et de contrôleurs pour la mise en œuvre d'Instruments de Musique Numériques (IMN). Malgré le grand nombre d'appareils et de contrôleurs sans fil disponibles, il y a eu peu d'évaluations systématiques de ces choix, lors de la conception d'IMN. La diversité de choix peut créer de la confusion lors de la sélection des interfaces. Ces interfaces résultent de la combinaison de périphériques physiques (matériel) et de protocoles de transfert de données (logiciel). Plus précisément, les performances en termes de latence (la réactivité d'un instrument et son évolution, lorsque des périphériques sont ajoutés dans un ensemble) et d'utilisabilité (la facilité d'intégration avec l'application de l'utilisateur et de l'utilisatrice) ne sont pas souvent spécifiées. Cette thèse présente des réponses à ces questions, en examinant deux problématiques principales : une analyse empirique de la latence dans les interfaces sans fil et la définition d'outils pour augmenter l'utilisabilité des IMN. Un système de mesure de latence de bout en bout est mis en œuvre pour mesurer les performances de différentes configurations d'interfaces de capteur sans fil. Ce système comporte une configuration d'interface de capteur, intégrée dans un système qui répond à une entrée d'activation de test et synthétise une sortie audio. Le Wi-Fi, le Bluetooth et le LoRa sont comparés aux interfaces filaires USB/série, avec Open Sound Control (OSC) et MIDI comme protocoles de transfert de données. Sur la base des résultats des travaux, les interfaces physiques fondées sur le Wi-Fi offrent la plus grande adaptabilité, en termes de nombre d'appareils simultanés, de bande passante globale, ainsi que de latence minimale, bien qu'il faille faire attention lors de l'appariement simultané de plus de cinq appareils. L'interopérabilité de l'interface dépendra du protocole de transfert de données: il y a un compromis à faire entre le besoin de flexibilité de représentation offerte par un standard plus ouvert comme OSC et le besoin de compatibilité avec les standards existants comme MIDI. La deuxième partie de cette thèse approfondit le concept d'interopérabilité et d'utilisabilité des interfaces de capteurs, du point de vue des cadres, outils et environnements d'appariement. Des recommandations sont proposées pour l'incorporation des interfaces de capteurs caractéristiques des utilitaires d'appariement visuel.

Acknowledgements

First and foremost, I would like thank my advisor, Marcelo Wanderley, for making this wonderful research experience possible and providing the support and guidance throughout my time at McGill. The positive impact of his work at Input Devices and Music Interaction Laboratory (IDMIL), Music Tech, Schulich School of Music and the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT) community was one of the primary reasons that I came to McGill in the first place. I would also like to thank Axel Mulder of Infusion Systems for providing the wonderful collaboration opportunity at I-CubeX that centered around the NSERC/FQRNT Industrial Innovation Scholarship, but continued beyond the official timeline through side projects and coffee table conversations.

To the fellow colleagues at IDMIL: Eduardo Meneses, Alex Nieva, John Sullivan, Geise Santos, Filipe Calegario, Carolina Medeiros, Anésio Azevedo, Ivan Franco, Travis West, Christian Frisson, Jeronimo Barbosa, and Darryl Cameron: it was a pleasure working with you in the lab, and hanging out with your families outside (and sometimes inside) it! For the long-time libmapper developers, Joseph Malloch and Steven Sinclair, I deeply appreciate all the in-person and online discussions we've had surrounding the framework, and hope that I will be able to contribute to the project long after my time at IDMIL, as the two of you have continued to do so.

I would like to express my deepest gratitude to my wife, Vivian, with whom I was able to share this unforgettable journey of moving into a new city, raising two lovely kids, Johanne and Sébastienne, while working on our PhD studies. Special thanks to my parents, Shiyi and Ann, and the in-laws, Scott and Juliane, for the support that has enabled all of this. To our dear friends, Stéphanie Akré and Thomas Gaudy: Thank you for the wonderful moments we shared over the past few years (and also the translation of the abstract!)

Last, but not least, thanks to Philippe Depalle for being the meticulous Second Reader of this thesis.

Preface

The inspiration for this work started in 2014 when supporting a project involving 10 iPod Touch devices that were used by the audience to control virtual avatars projected on the side of a building as part of a digital interactive artistic installation. While we were frantically trying to work through various technical and logistical challenges often associated with such projects to get something that "just works", I was quite interested in trying to figure out exactly *how many* devices a particular channel could practically support in such a situation, and *how fast* each device could transmit. Given the time constraints of the project, I was unable to further explore the issue after we established a functional network configuration for the event. Since then, I have been exposed to various other projects that make use of wireless sensor interfaces including many designed by colleagues at the Input Devices and Music Interaction Laboratory at McGill such as the T-Stick (Malloch and Wanderley 2007), Vibropixels (Hattwick, Franco, and Wanderley 2017), GuitarAMI (Meneses, Freire, and Wanderley 2018), Prynth (Franco and Wanderley 2016a) and Probatio (Calegario et al. 2017). Speaking with the creators of these devices and systems, and studying related literature in the New Interfaces for Musical Expression (NIME) community, it appears that generally in most cases a similar approach of "it works adequately for our application" is applied, and often any extended questions such as "could we support a T-Stick duo or quartet at the current sensor transmission rates and message sizes?" remain as an exercise for the future. Often, these unforeseen limits can lead to significant design changes and unexpected additional development. Moreover, while we see an increasing number of wireless interfaces in the academic and commercial realm, the actual performance metrics are often not specified, and there is little work to provide a standardized testing and comparison of these interfaces. In this work, I hope to achieve a better understanding of the practical limits of wireless interfaces that can lead to more informed decisions when building these systems or selecting devices. While the work described pertains mostly

to interfaces used for musical interaction, the results are applicable to general interactive systems where distributed, wireless connections between sensors and receiving devices are used, and the overall methodology applies to sensor and device networks in the Internet of Things community as well.

The work in this PhD thesis is motivated by my background in classical musical performance combined with formal education in electrical and computer engineering, leading towards a number of interdisciplinary projects in the field of new media. The work performed in the PhD thesis continues through the building of tools and platforms to support the process of developing novel ways of interacting with music and media. Part of the work was performed in collaboration with an industrial partner, Infusion Systems, as part of an NSERC Industrial Innovation Scholarship for the first three years of the research.

Contributions

The work described in this thesis lead to a number of peer reviewed (first author) publications at international conferences describing the evaluation of latency and mapping tools in the context of Digital Musical Instrument (DMI) design. The evaluation system is also used in a number of collaborative context (second author), and the overall experience is also applied in larger projects to inform the general process of DMI and framework development.

During the PhD the author also supported a number of artistic works involving the development of connectivity and mapping tools, specifically in the context of pipe organ interfacing. Additionally, as part of an industrial collaboration, commercial products were also conceived, prototyped, and evaluated, and prepared for manufacturing.

The remainder of this section provides an overview of these contributions, which are described in further detail in their respective context in the thesis.

Publications (first author)

- **Wang, J**., Meneses, E., Wanderley, M.M. (2020). The Scalability of Wi-Fi for Mobile Embedded Sensor Interfaces. Birmingham, U.K., In Proceedings of the Conference on New Interfaces for Musical Expression (NIME) (Described in Chapter 4.1).

- **Wang, J**., Malloch, J., Sinclair, S., Wilansky, J., Krajeski, A., Wanderley, M.M. (2019). Webmapper: A Tool for Visualizing and Manipulating Mappings in Digital Musical Instruments. Marseille, France, 14th International Symposium on Computer Music Multidisciplinary Research (Described in Chapter 5.2).

- **Wang, J**., Mulder, A., Wanderley, M.M. (2019). Practical Considerations for MIDI over Bluetooth Low Energy as a Wireless Interface. Porto Alegre, Brazil, In Proceedings of the Conference on New Interfaces for Musical Expression (NIME) (Described

in Chapter 4.2).

- **Wang, J**., Pritchard, B., Nixon, B., Wanderley, M.M. (2017). Explorations with Digital Control of MIDI-enabled Pipe Organs, Sound and Music Computing (SMC).

- **Wang, J**., Malloch, J., Huot, S., Chevalier, F., Wanderley, M.M. (2017). Versioning and Annotation Support for Collaborative Mapping Design, Sound and Music Computing (SMC) (Described in Chapter 5.2).

Publications (other)

- Santos, G., **Wang, J**., Medeiros, C., Wanderley, M. M., Tavares, T., and Rocha, A. (2021). Comparative latency analysis of optical and inertial motion capture systems for gestural analysis and performance. In Proceedings of the Conference on New Interfaces for Musical Expression (NIME) (Described in Chapter 4.5.2).

- Calegario, F., Tragtenberg, J., **Wang, J**., Franco, I., Meneses, E.A.L., Wanderley, M.M. (2020). Open Source DMIs: towards a replication certification for online shared projects of digital musical instruments. Copenhagen, Denmark, In Proceedings of the International Conference on Human-Computer Interaction (HCII).

- Calegario, F., Wanderley, M. M., Tragtenberg, J., **Wang, J**., Sullivan, J., Meneses, E., Franco, I., Kirkegaard, M., Bredholt, M., Rohs, J. (2020). Probatio 1.0: collaborative development of a toolkit for functional DMI prototypes. In Proceedings of the Conference on New Interfaces for Musical Expression (NIME).

- Meneses, E., **Wang, J**., Freire, S., Wanderley, M.M. (2019). A Comparison of Open-Source Linux Frameworks for an Augmented Musical Instrument Implementation. Porto Alegre, Brazil, In Proceedings of the Conference on New Interfaces for Musical Expression (NIME), (Described in Chapter 4.5.1).

- Nieva, A., **Wang, J**., Malloch, J., Wanderley, M.M. (2018). The T-Stick: Maintaining a 12 year-old digital musical instrument. Blacksburg, Virginia, In Proceedings of the Conference on New Interfaces for Musical Expression (NIME).

Artistic Performances and Collaborations

- 2020-21: Implementation of the Global Hyperorgan: a technical realization of remote connected mapping.
- 2019: Cross-Canada performance of a Bach Prelude on two Casavant Organ Consoles. Church of St. Andrew and St Paul (Montreal) and Pacific Spirit United Church (Vancouver).
- 2018: An Evening of Works for the Augmented Pipe Organ: George Rahi MFA concert.

Industrial Collaboration

I-CubeX Pi-Shield

During the exploratory phase of the industrial collaboration, the author lead in the design and manufacture of an add-on sensor interface shield for the Raspberry Pi[1] family of embedded computers (Figure 1). This allowed the extensive catalogue of I-CubeX sensors to be used with the large RPi community. Along with hardware design, the author also built various software interface drivers and example applications for a variety of programming environments on the Raspberry Pi, and created supporting documentation, promotional materials, and managed the marketing for a successful international crowd-sourcing campaign[2].

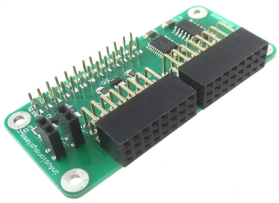


Fig. 1 The I-CubeX PiShield.

Fig. 2 The I-CubeX WiDig.

I-CubeX Wi-Dig

A new Bluetooth Low Energy / Wi-Fi addon board was designed to add wireless connectivity to the existing I-CubeX Universal Serial Bus (USB) digitizers, resulting in the world's first customizable, MIDI over Bluetooth Low Energy (BLE-MIDI) capable sensor interface with built-in sensor input to Musical Instrument Digital Interface (MIDI) message mapping (Figure 2). The author was responsible for the hardware and firmware design, prototype production, testing, preparation for manufacturing, and user documentation.

[1] https://www.raspberrypi.org/

[2] https://www.kickstarter.com/projects/infusion/pishield-sensor-interface-board-for-raspberry-pi

List of Acronyms

ADC Analog-to-Digital Converter

ALSA Advanced Linux Sound Architecture

AMI Augmented Musical Instrument

API Application Programming Interface

AR Augmented Reality

ASIO Audio Stream Input/Output

ASCII American Standard Code for Information Interchange

BLE Bluetooth Low Energy

BLE-MIDI MIDI over Bluetooth Low Energy

CCRMA Center for Computer Research in Music and Acoustics

CIRMMT Centre for Interdisciplinary Research in Music Media and Technology

DAC Digital-to-Analog Converter

DAT Digital Audio Tape

DAW Digital Audio Workstation

DHCP Dynamic Host Configuration Protocol

DMI Digital Musical Instrument

ECDF Empirical Cumulative Distribution Function

HCI Human Computer Interaction

IC Integrated Circuit

IDMIL Input Devices and Music Interaction Laboratory

IEEE Institute of Electrical and Electronics Engineers

IoMT Internet of Musical Things

IoT Internet of Things

IMU Inertial Measurement Unit

IP Internet Protocol

IR	Infra-Red
ISM	Industrial, Scientific and Medical
LED	Light Emitting Diode
MIDI	Musical Instrument Digital Interface
mocap	Motion Capture
NIME	New Interfaces for Musical Expression
NSERC	Natural Sciences and Engineering Research Council of Canada
OS	Operating System
OSC	Open Sound Control
OSI	Open Systems Interconnection
PAN	Personal Area Network
PWM	Pulse Width Modulation
PCB	Printed Circuit Board
QoS	Quality of Service
RAM	Random Access Memory
RF	Radio Frequency
RTP	Real-Time Protocol
SMT	Surface Mount Technology
SLIP	Serial Line Internet Protocol
SPP	Serial Port Profile
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
UDP	User Datagram Protocol
VR	Virtual Reality

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This introductory chapter establishes the overall context of the thesis by first presenting an overview of DMIs wherein the work is situated. Then, the conceptual components that make up a DMI are presented to reveal the relevance of the core contributions of the thesis that focus on sensor interface protocols for wireless interfaces. The lack of performance and usability evaluation in existing literature regarding wireless interfaces in the context of DMIs is introduced, followed by the presentation of a specific series of questions along with a proposed approach for answering them through empirical evaluations.

1.1 Digital Musical Instruments

Using computer and digital input devices, new musical interfaces can be produced to control sound parameters in many different ways, and allow new ways of interacting with the production of sound and musical output ranging from the real-time control of audio parameters, to the manipulation of higher-level compositional structures (Rowe 1992). Collectively

known as DMIs (Miranda and Wanderley 2006), the study, development, and usage of these devices and systems open up novel avenues of interdisciplinary research that leads to new technological developments, tools for research, artistic works, and commercial products.

More generally, such devices and systems fall within the realm of interactive digital media, where some kind of sensor input is used to facilitate user manipulation of processes that control the production or manipulation of media on a computer (Pennycook 1985). The sensors could be any kind of input device that captures and translates some aspect of a physical phenomenon in the real world into electrical signals that are then digitized for computation (Fraden 2010). Some examples of such systems, beyond musical instruments, include user-influenced visual art installations, and interactive digital environments that respond according to any kind of sensor input. One key feature of such devices and systems is that, unlike acoustic instruments or physical media, the relationship between input and output are no longer constrained by the nature of physical mechanisms alone. Computer algorithms can arbitrarily associate an input and output parameter, and provide limitless relationships between the two (Bongers 2000). For the remainder of this thesis, we will refer to DMIs specifically given the context of the research, but with an understanding that the concepts can be generalized to any digital interactive system that requires sensor input, where the output of interest is not limited to the control of musical parameters.

One conceptual representation of a DMI is the "three-layer" model (Hunt and Wanderley 2002), shown in Figure 1.1. Here, the input consists of sensors and sensor interface that transduce a signal relating to a physical phenomenon, in this case, a musical gesture of interest, into numerical data. It is then fed through a mapping layer that translates the input signals into synthesis parameters, that finally feeds into a synthesis system that produces digital audio which can be played back using a Digital-to-Analog Converter (DAC). Some examples of DMIs developed at IDMIL and CIRMMT are shown in Figure 1.2.

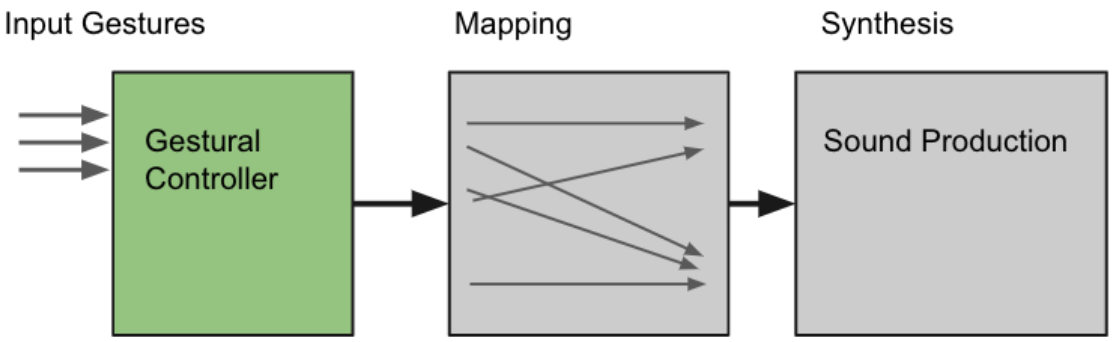
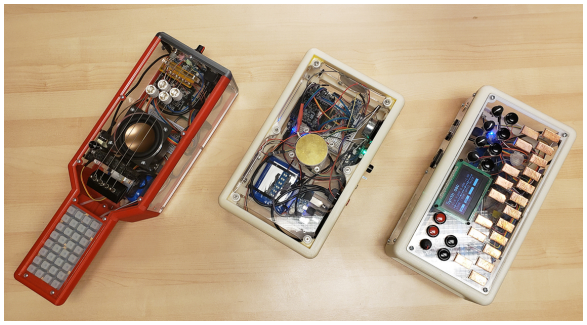
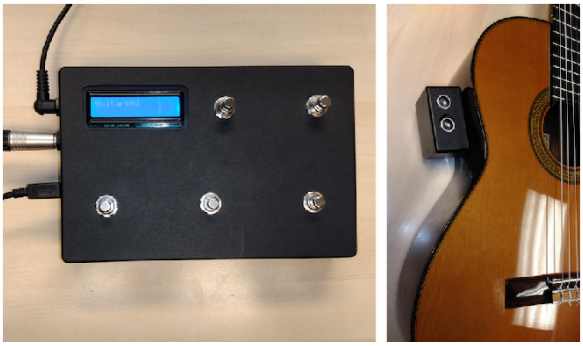


Fig. 1.1 The "three-layer" conceptual model of DMIs.

(a) Franco's "The Mitt".



(b) Sullivan's "NoiseBoxes".



(c) Meneses' "GuitarAMI".

(d) Malloch's "T-Stick" (Photo by E. Meneses).

Fig. 1.2 Some examples of DMIs, images by respective authors unless otherwise noted, and reproduced with permission.

During implementation, the flexibility of digital systems allows these components to be distributed or integrated accordingly with either internal or external connections, depending on the design constraints and physical limits of the components selected. In our modern digital age, increasingly capable yet affordable and accessible electronics and sensing technologies, computational platforms, and software ecosystems provide limitless options to realize the design and implementation of DMIs.

1.2 Controllers vs. Instruments

To further clarify scope of the presented work in this thesis, it is necessary to make the distinction between a *controller* and an *instrument*. Since modern general computing devices can take many physical forms with different levels of core and peripheral configurations (e.g. desktop computers with detached input/output devices compared to laptops and tablets), so can a DMI. Figure 1.3 shows four devices, a keyboard MIDI controller, a T-Stick (Malloch and Wanderley 2007), The Mitt (Franco and Wanderley 2016b), and keyboard synthesizer. The MIDI controller, as well as the T-Stick, do not possess the ability to generate sound but instead send control signals to an external synthesizer. The Mitt, like the keyboard *synthesizer*, on the other hand, contains the mapping and synthesis components, and is consequently able to produce sound from within. At the same time, these instruments can also send control signals as well, and function as controllers to be operated with another synthesis system, if desired. The process by which the T-Stick as an interface or controller was developed into a musical instrument through stages of mapping and synthesis design (Malloch and Wanderley 2007) provides an exemplary process for the realization of a DMI. Various physical updates to the T-Stick has been made over the years (Nieva et al. 2018), the device has supported research on mapping (West et al. 2021), and

the creation of various compositions (Stewart 2009; Fukuda et al. 2021).

Some examples from the commercial realm include the wind-instrument-inspired interfaces such as the Sylpho[1], Akai EWI[2], Yamaha WX-series[3] controllers, compared with the Casio Digital Horn or Roland Aerophone[4] instruments. The former are controllers that transmit button presses and breath pressure, musical gestures, to another synthesis device but do not produce sound by themselves, while the latter contain sound synthesis and production components on-board.

In this work, we will focus the attention to controllers while acknowledging the fact that certain standalone instruments exist. The controllers, from a technical implementation perspective, can be considered more generally as sensor interfaces where the sensors are selected to capture specific signals of interest associated with gestures for musical input.

1.3 Wireless Musical Controllers and Latency

A key development with the growth of mobile computing devices, is the near ubiquitous nature of wireless technology (Bi, Zysman, and Menkes 2001). Supported by improved power efficiency and robust battery solutions demanded by increasingly powerful devices in the mobile phone and portable computing market, most computational devices have some mode of wireless operation. Wireless is appealing for musical controllers as it allows the implementation of untethered devices, in the same manner that most traditional acoustic instruments are not confined by wiring.

Traditionally, wireless interfaces have been deemed as unreliable, due partially to the limitations of battery reliability and wires were often recommended where possible in mu-

[1] https://www.aodyo.com/presentation-sylphyo-page-en-3.html
[2] https://www.akaipro.com/products/ewi-series
[3] https://usa.yamaha.com/products/music_production/midi_controllers/wx5/index.html
[4] https://www.roland.com/ca/products/aerophone_ae-10/

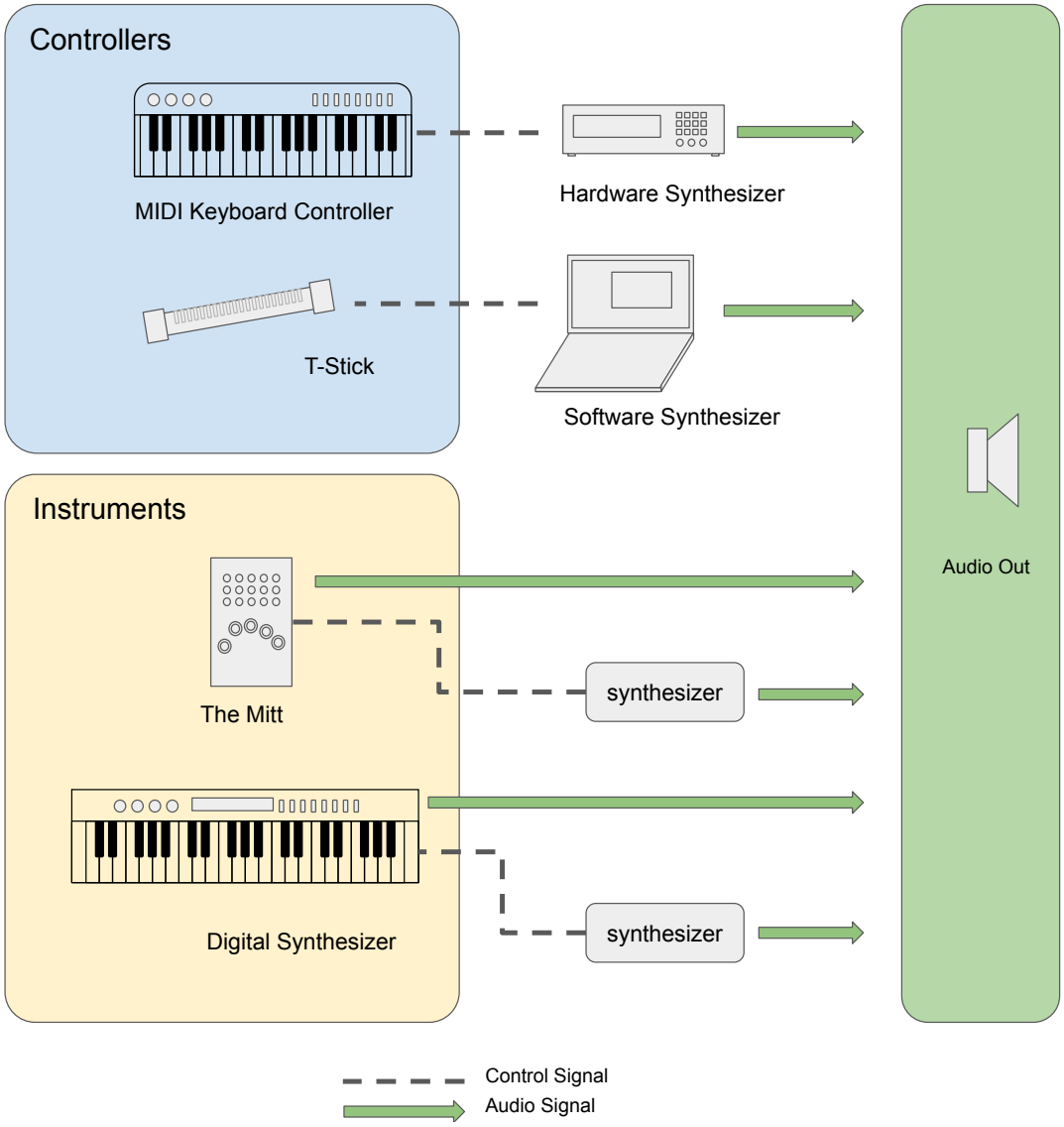


Fig. 1.3 Controllers rely on another components for the synthesis and production of sound, while digital instruments can synthesize sound as well as emit control signals, depending on mode of operation.

sical instrument contexts (Cook 2001), where performance reliability is deemed a critical factor (Buxton 1997). However, the improvements in wireless and battery technology have made their use more feasible for DMIs (Cook 2009). In addition to reliability, latency is yet another issue introduced by wireless interfaces. The notion of latency in this context pertains to how quickly a sensor signal associated with a physical event, such as a musical input gesture, is transmitted by the sensor interface into the system, and subsequently translated into output sound. This affects the responsiveness of the instrument, and wireless interfaces introduce additional latency in the transmission process.

Acoustic instruments, for example, have nearly instantaneous physical response from the moment of actuation. One exception is the low register of large wind instruments such as the tuba or pipe organ where some time is needed before the initial source excitation and the effective movement of the air mass that lead to audible sound waves. Beyond such initial delays, the main source of delay is due to acoustic propagation caused by the distance between the sounding mechanism and the listener position. For instruments where the sound producing mechanism is within a metre or so from the ear, this delay is a few milliseconds at most or a few tens of milliseconds in a chamber music setting. However, an extreme case is that of a pipe organ in a church, where the distances between the console, near, and far pipes can be tens of metres or more. In this situation the propagation latency can be over 100 ms and requires unique adaptations by the organist to synchronize the sound from different sets of pipes and other musicians in the space. Finally, when dealing with networks and the potential for remote collaboration in the context of Telematic Music (Oliveros et al. 2009), at global distances even the speed of light becomes a factor, and tens of milliseconds can be added even assuming a direct wire connection between two end points across the globe (Carôt and Werner 2009). In real life networking implementations, various routing and switching hardware will contribute to much larger latency values (Jiang

et al. 2018).

Unlike in the general field of Human Computer Interaction for contexts such as web browsers and productivity applications where a latency of about 100ms have been considered as "real-time responsive" (Nielsen 1994), for the context of real-time musical interaction, the "gold-standard" value is an order of magnitude less, at 10ms (Moog 1986; Wessel and M. Wright 2002). However, the exact value of what is considered as acceptable latency often depends on the musical context. Latency requirements in various musical contexts revealed a range of a few milliseconds up to around 200 ms (Rottondi et al. 2016). In a live-monitoring situation, the acceptable latencies were found to be anywhere between 1.2 and 42 milliseconds (Lester and Boley 2007). For a non-haptic instrument, the Theremin, a study showed a just-noticeable-difference of 20-30 milliseconds (Mäki-Patola and Hämäläinen 2004). Strategies for coping with different amounts of latency using various adaptive musical collaboration methods are presented by (Carôt and Werner 2009). In the context of sensor interfaces, we focus on the individual response of an individual device as it provides the baseline signal transmission pipeline that may be used for any of these above settings.

The responsiveness of an instrument is important for the expressive potential, or "Control Intimacy" of a musical interface (Moore 1988). The three main problems associated to "Control Intimacy" are: *performance capture*, or how well the system is able to sample the input gesture of interest, *synthesis control*, or how the control signals can be used to cause immediate and appropriate responses in the produced sound, and *control transmission*, which relates to how effectively medium between the sensing mechanisms and the synthesizer is able to transport the necessary data in real-time. The work surrounding latency measurements in this thesis focuses on a portion of the third problem in the context of wireless sensor interfaces.

One other issue related to timing is not only the extent of the delay (latency), but the

differences in delay between successive events (jitter). Existing literature have suggested jitter values of around 1 to 1.5ms to be acceptable, and further research on varying the amount of jitter (Jack et al. 2018) confirm that the perception of audio quality can be impacted with even small amounts of jitter but may also be context dependent. Therefore, any kind of evaluation of the latency in these systems should be accompanied by an awareness of jitter as well, including the absolute amount of deviation between maximum and minimum values, as well as the spread. It is possible to compensate for jitter by adding a timing buffer up to the maximum amount of expected difference (Brandt and Dannenberg 1998). Of course, this compromise will increase the amount of overall latency, and would not cover exceptional cases where the jitter exceeds the expected value. In general, because wireless mediums have the possibility to drop packets or experience long delays, it may be necessary to implement a timeout where values beyond a certain latency are simply ignored, or a negative acknowledgment be sent which may trigger retransmission, such as the case implemented by certain networking protocols such as Transmission Control Protocol (TCP) (Stevens 1997). Regardless of the specific mechanism implemented to handle these situations, the important consideration is that an analysis of the latency in such systems should be accompanied by an awareness of associated jitter values, and presented as part of a device's performance metrics in the documentation.

Having established the overall context, we approach the remainder of the work with the overall goal that DMIs and musical controllers should exhibit as little latency as possible, and the final section in this introductory chapter describes in more detail the primary investigations that are performed in the work to investigate the responsiveness of wireless interfaces. In addition to the primary evaluation of latency performance, the latter portion of the work will explore the sensor interfaces from the perspective of usability and compatibility perspective, and present the findings within the context of mapping tools and

frameworks in the context of DMI design.

1.4 Method and Scope of Contributions

Despite the growing number of wireless devices and controllers used for DMIs, there is little systematic evaluation of their performance. Specifically, the delays introduced by the wireless interface impacts responsiveness of the system, often significantly more than their wired counterparts (McPherson, Jack, and Moro 2016). Additionally, there are multiple potential protocols that could be used, and it is not always obvious which protocol may be suitable for a given application. In this work, after reviewing the literature on existing implementations of wireless musical controllers and the evaluations performed on them as described in existing literature in Chapter 2, we perform a number of evaluations on wireless sensor interfaces that attempt to answer the following questions:

- *What are the minimal possible latencies provided by a particular wireless interface?* This is a useful quantity as an interface that exceeds a certain threshold may deem it unsuitable for a specific application that requires a certain degree of responsiveness.
- *How fast/often can a given wireless interface transmit?* This metric is useful as certain gestures of interest may need to be sampled and transmitted at a certain rate to capture and translate the appropriate level of control for a particular application.
- *How many wireless interfaces can operate for a given network configuration?* For collaborative context where multiple devices are used, this determines the number of instruments that can be used at the same time.
- *What is the most suitable protocol for a given DMI configuration, and how can we incorporate our findings into existing tools?* This is a more general question that can

inform DMI design activities.

We select protocols and configurations to be tested among a much larger list of possible solutions, after explaining the selection criteria while considering their appropriateness in the context of musical applications. When approaching these choices, we focus on a wireless interface from the perspective of both the physical hardware interface along with the communication protocol, which will be introduced in more detail in Chapter 2. We then present the system that was implemented to evaluate and compare protocols in Chapter 3, by building sensor interface prototypes that minimally implement and thus represent each protocol of interest to be tested. In Chapter 4 a number of case studies where the evaluation technique is applied for a specific application as well as the relevance of the contribution to the particular context is presented, including the effects of device scaling for Wi-Fi (Chapter 4.1), comparison between MIDI over BLE, OSC and MIDI over Wi-Fi (Chapter 4.2), a three-way comparison between Wi-Fi, BLE, and LoRa (Chapter 4.3), and BLE-scaling (Chapter 4.4). Some collaborative projects are presented in Chapter 4.5 (resulting in second author publications) where the test methodology was adapted for other settings including embedded Linux platforms, and motion capture systems. In Chapter 5, we investigate the usability aspects of sensor interface protocols from the perspective of compatibility and present the relevance of our evaluation on the work involving mapping tools and frameworks, and apply the findings of the thesis on an existing instrument. Finally, in Chapter 6 we summarize findings from the evaluations presented in previous chapters and provide discussion on the implications, limitations and extensibility of the work presented.

The work described in this thesis takes place within the field of Music Technology, where creative processes and results in addition to or alongside scientific research. Two

specific approaches, *practice-led* and *practice-based* are common methods in which such research takes place (Candy 2006). The work presented in this thesis can be considered as *practice-led*, as it involves the development of tools and approaches to aid a particular artistic process, rather than the artistic products themselves. While the main contribution of the work as demonstrated by the case studies target DMIs and musical controllers, the overall process of evaluating devices can be generalized to other applications. Of course, such applications may require less or more stringent demands on performance depending on the context. For example, an interactive installation that responds to high-level user activity or presence may reply on slower time-varying signals and may not require as strict latency specifications, while others such as Virtual Reality (VR) or Augmented Reality (AR) applications (Maier et al. 2016), can be even more demanding than some musical applications.

The process and outcomes described in this thesis include work performed at the IDMIL and CIRMMT at McGill, along with the outcomes of a 3-year collaboration with Infusion Systems as part of an Natural Sciences and Engineering Research Council of Canada (NSERC) Industrial Innovation Scholarship. Infusion Systems have been providing sensor interfacing solutions for digital interactive arts for the past 20 years, and one of the main goals of this collaboration is to further refine their existing wireless interfaces as was thus a fitting partner for this research. Sections of Chapter 4.2 describe the outcomes of this collaboration.

Chapter 2

Related Work

In this chapter, we present existing work in the literature relevant to our subsequent analysis. First, we present a brief overview of the nature of the signals in the context of DMIs. Then, we break down sensor interfaces with a granularity relevant to our subsequent evaluations, identified by two main components: the physical hardware interface and communications protocol. From there, example interfaces in the literature are presented in the context of these two aspects. Then, we describe previous attempts to evaluate the latency behaviour of DMIs. Finally, we present the metrics that are considered by this thesis in the context of their potential application for DMIs.

2.1 Overview of Signals for DMIs Applications

Signals specific to the context of DMIs are associated from physical events arising from input gestures. Ways of selecting the appropriate input sensor for a particular gesture is an interesting problem that overlaps the field of Human Computer Interaction (HCI) (Card,

Mackinlay, and Robertson 1991), and many tools and approaches there can be applied to DMI research (Wanderley and Orio 2002). There are many potential interaction paradigms for DMIs (Birnbaum et al. 2005) that will dictate what kind of signals may be captured for a particular application, and whether exact time-synchronization is required or not, and whether the signal is discrete (made up of individually separate events in time) or continuous (constantly evolving over time) is sufficient from the perspective of sampling and transmission. From our perspective of latency performance and analysis, we are more interested in the latter. Table 2.1 shows the characterization of signals both in time and value, that are relevant to sensor interfaces from a DMI context. Note that for completeness, we also show a third column of whether the actual signal *values* are discrete or continuous as well in addition to how they occur *in time*. A final, perhaps obvious note should be made here that, at the end of the day, for digital systems everything will be *discretized* for actual storage and retrieval.

Event	**Time**	**Value**
Note on (key)	Discrete	Discrete
Slider	Continuous	Continuous
Breath pressure	Continuous	Continuous

Table 2.1 A non-exhaustive list of different types of events and gestures in the context of DMIs.

The consequence of this distinction, is that sensor interfaces may operate either by transmitting a message only when a specific event has occurred, or constantly in "streaming" mode depending on the application (Moore 1988).

2.2 Wireless Sensor Interfaces

The data transmission component of a sensor interface is facilitated by a communications system involving many layers from the physical transmission of electrical or Radio Frequency (RF) signals through a medium (such as cables or the air, respectively), to various supporting protocols that support the packaging and delivery of the data. Both the receiver and transmitter must employ the matching system in order to inter-operate. The Open Systems Interconnection (OSI) Model for networking is a widely used method to characterise modern digital networks (Zimmermann 1980). In the context of DMIs implementations, it is often neither necessary nor practical to delve into low level details of the networking stacks involved, and in our analysis we present a simpler layered model consisting of fewer layers, as shown in Figure 2.1 which contains both the original OSI model and the simplified version with just two layers used in our subsequent analysis and discussion.

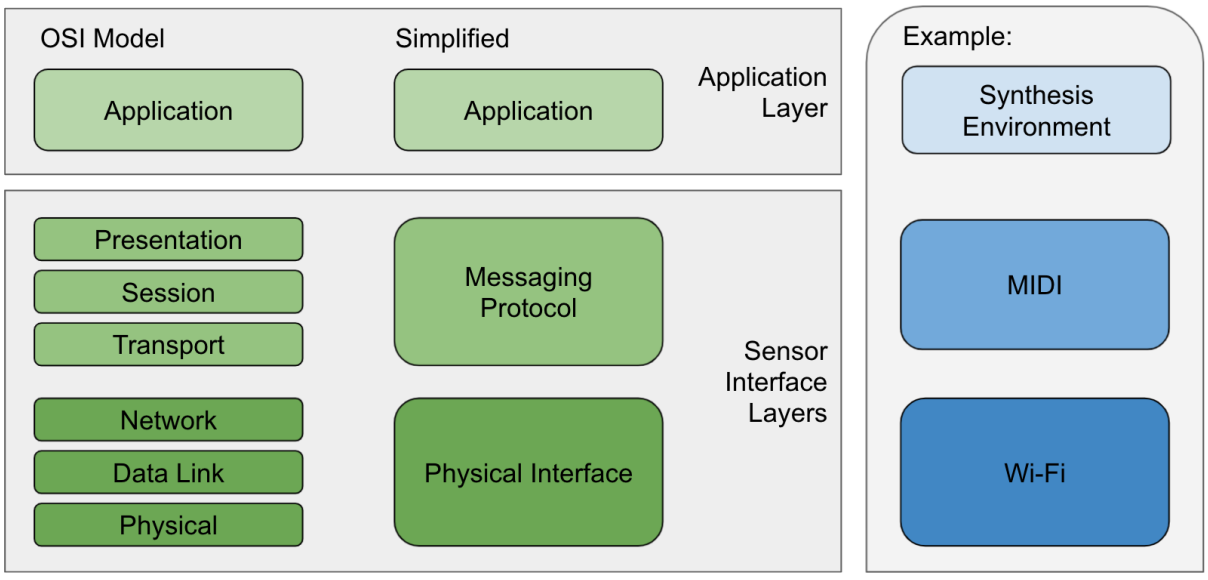


Fig. 2.1 A simple layered model showing the relevant components of a sensor interface (centre) alongside the OSI model (left), with an example sensor interface implementation (right).

In the reference OSI model there are seven layers that describe the underlying structure of the networking stack, starting from the physical layer at the bottom that defines how the signals are physically propagated, to various supporting protocols above that provide the transmission and delivery mechanisms. From the perspective of a DMI designer, however, it is often not the goal to develop or optimize a particular networking stack but rather select the most appropriate configuration from a selection of potential candidates.

From an application perspective, where DMIs are implemented using commodity hardware (i.e., consumer computing devices such as desktop/laptop computers, smartphones and tablets), there may also be practical limits in terms of the physical hardware interfaces supported due to the availability of built-in radio hardware and drivers. In the context of this research, these limits then provide an opportunity to narrow down the selection criteria for our evaluations and focus on specific interfaces due to their existing support within systems that are used by the DMI community. However, it does not preclude the possibility of adding new hardware to provide support for physical interfaces that are not natively supported by commodity computing devices, and some of our subsequent examples presented show less widely supported RF interfaces being employed in some DMI contexts as well.

Moving on, the two relevant layers distinguished here serve two crucial and related roles. First, the hardware interface layer provides the necessary support including antennae, RF electronics, and radio control firmware to physically transmit the signals over the air. Then, the messaging protocol defines the software specifications that determines how the data is presented and transmitted. Following we present examples of some hardware interfaces and messaging protocols that are evaluated in the remainder of this work.

2.2.1 Hardware Wireless Interfaces

There are two relatively ubiquitous hardware wireless interfaces on modern computing devices: Wi-Fi and Bluetooth. Nearly every mobile computing device on the market today will support these two wireless protocols, and as such they are the most obvious starting choices for the implementation of a sensor interface since it requires the least amount of additional hardware and system drivers required for the host operating system. Following is a brief overview of these two technologies. In general, Wi-Fi has higher transmission speeds and bandwidth compared with Bluetooth.

Wi-Fi

Wi-Fi is a number of related wireless protocol standards defined under the umbrella of IEEE 802.11 (Gast 2005). These standards define physical transmission protocols that are implemented on devices that are designed to communicate with each other, and include consumer-grade computational devices within the home and office environments. For many desktop and laptop workstations, Wi-Fi is the default interface for network connectivity as an alternative to wired Ethernet connections, and as such provides relatively fast transmission speeds and high bandwidth. It operates in both the 2.4GHz and 5.8GHz Industrial, Scientific and Medical (ISM) bands, which is a portion of the radio frequency spectrum designed for devices that can be operated by users without professional radio operating licenses, which make up most consumer grade electronic devices.

Wi-Fi operates at a wide variety of bandwidths depending on the version of the implementation, but the tested versions make use of the 802.11G operating at 54 MBps.

Bluetooth and BLE

Bluetooth is a short range RF protocol designed to connect low-power mobile devices as part of Wireless Personal Area Networks (PANs) operating in the 2.4GHz ISM band. It is defined by the Institute of Electrical and Electronics Engineers (IEEE) 802.15.1 specification, and has undergone a number of revisions since the first version in 2001 (Bhagwat 2001). Designed for low power wireless devices and accessories, Bluetooth is used to enable wireless connection of audio headsets, input devices, and even file transfer between mobile devices.

In this work we primarily focus on Bluetooth Low Energy, which was released part of version 4.0 of the Bluetooth protocol (Gomez, Oller, and Paradells 2012), as it is supported by most devices on the market today and a number of commercial musical interfaces make use of MIDI over BLE for connectivity.

BLE has a physical bandwidth of 2 MBps, but it has relatively long connection interval which are windows in which transmissions can take place, and 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 (Wessel and M. Wright 2002) 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 at *most* 15ms (MMA 2015). Later in practical measurements we see the connection interval, as tested on the Mac Operating System, is in fact fixed at the higher interval of 11.25 ms. 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.

Other RF Interfaces

While Bluetooth and Wi-Fi enjoy nearly ubiquitious hardware support by most modern computing platforms, the same cannot be said of other wireless interfaces. However, a number of existing protocols including ZigBee, nRF2401, and LoRa have been employed for wireless sensor interface applications as well. However, the main limitation with these interfaces is the need for additional hardware as they are not normally supported by commodity computing devices. Of course, hardware adapters that provide compatible interfaces could be used, but they add another layer of complexity to the system and as a result, are not the primary focus of our evaluation. However, they can be evaluated in the same way assuming compatible interfaces or bridging adapters are available.

2.2.2 Messaging Protocols

In this section we describe two messaging protocols that are used in subsequent evaluations in this work. In general there are many potential options when a system is built from scratch, but to remain within the general context of DMI design, we choose protocols that are widely supported by existing applications. In this regard, MIDI and OSC are the primary two candidates of interest.

Musical Instrument Digital Interface

MIDI is one of the most widely accepted standards for interconnecting musical devices and controllers (Loy 1985). Consisting both of a physical wired interface specification as well as software protocol, MIDI compatible devices can inter-operate across a wide variety of hardware controllers, synthesizers, as well as software applications that implement Digital Audio Workstations (DAWs), synthesizers, and programming environments. On desktop

platforms, drivers associated with a MIDI device will present a specific sets of MIDI input and output ports to the operating system, and software can make use of the API provided by the operating system to access and send/receive data through the MIDI ports.

While the MIDI standard defines the communication protocol, the wired physical interface (and even the connectors used), in our analysis we mainly look at MIDI from the first definition. In this regard we could, for example, transmit MIDI over other physical interfaces (and especially wireless ones, in the context of our work).

MIDI is defined by a series of event-based messages that contain a status byte followed by data bytes (Lehrman and Tully 1993, Chapter 1). Table 2.2 shows some commonly used MIDI messages. The status byte encapsulate the type of message and the one of 16 channels the messages applies to (defined by the four CCCC bits), while the data bytes contain the actual value of the message. The status and data bytes are differentiated via the starting bit of '1' and '0', respectively. This also has the consequence of limiting the data byte to 7 bits.

Status	**Data**	**Data**	Message
`tttt cccc`	`0ddd dddd`	`none`	generic message **T** with single 7-bit data **D**
`tttt cccc`	`0ddd dddd`	`0ddd dddd`	generic message **T** with two 7-bit data **D**
`1001 cccc`	`0nnn nnnn`	`0vvv vvvv`	note **N** on with velocity **V**
`1000 cccc`	`0nnn nnnn`	`0vvv vvvv`	note off with velocity **V**
`1100 cccc`	`0ppp pppp`	none	program change **P**

Table 2.2 Examples of 2 and 3-byte MIDI messages.

One main feature, as well as limitation of MIDI is the strict standards with which messages are defined and represented. By using integer representations of 4 bits to represent the message type, channel number, and either 7-or 14-bit integer values provide a rather limited range of possible values (Moog 1986). While it is possible to encapsulate custom data in the form of a raw byte stream using System Exclusive (SysEx) messages, it requires

a parser on the other end to make sense of this raw data. While limiting, the stringent standard of MIDI means that any "MIDI compatible" device is capable of responding to a particular message, and devices can interoperate without any additional translation or mapping. By conforming to even more specific standards such as General MIDI (Lehrman and Tully 1993, Chapter 9), a standard program bank of instruments will allow devices to follow the same instrument list and know exactly what an instrument is referred to via a particular Program Change value.

However, the trade-off is that there is little flexibility in representation beyond the fixed resolution format and relatively low transmission rates based on the 1980's technology at the time. The limitations of MIDI were noted early on (Moore 1988), and these limits have been reiterated over time (Rowe 2005) despite its widespread uptake by the community (Igoudin 1997), supported by nearly universal adoption by hardware and software manufacturers. The new MIDI 2.0 standard attempts to address these issues through features like increased resolution, faster transmission rates, and additional flexibility of representation (Lehrman 2020). One key feature is that to maintain the interoperability of the original standard, MIDI 2.0 is also designed to be backward compatible and will revert to the old standard when needed. However, currently there are few devices that support this emerging standard that can make advantage of these new features. From the perspective of DMI design, while it is not necessary to conform to the limiting standards of MIDI, it is still often useful (or necessary) to interface with commercial software and hardware that use MIDI exclusively as the communications interface.

Open Sound Control

Open Sound Control is a messaging format designed to carry real-time control data over general-purpose networks (M. Wright 2005). Developed with the availability of higher speed

Ethernet, OSC attempts to address some of the limitations of the MIDI protocol in interactive media settings. Unlike MIDI that was designed for a specific physical interface (31500 baud serial connection), OSC was designed to be "transport-agnostic" and can operate at the speed of the networking interface used and can be on the order of gigabits per second or more on modern wired networking devices. To send and receive OSC, a User Datagram Protocol (UDP) server socket is opened on the receiver end and the transmitter then sends to this destination Internet Protocol (IP) address and port via standard UDP messages. The increase in expected bandwidth allows for a less constrained messaging specification that allows symbolic, human readable names to be used for parameters along with longer numerical representations. Essentially, beyond the basic messaging specification that dictates a base address in the form of a string and another string for each value name, the actual data can be either a number of higher width numerical values (32 and 64 bit integers and floating point values), strings, or even raw bytes. The format is described in more detail by (Freed 1997), and Figure 2.3 shows some example OSC messages.

address '/**addr**' and one Int32 value (1000)	**Notes**

`0x2F (/) 0x61 (a) 0x64 (d) 0x64 (d)`
`0x72 (r) 0x00 ( ) 0x2C (,) 0x69 (i)`
`0x00 ( ) 0x00 ( ) 0x03 ( ) 0xE8 (è)`
	'i' suggests single int type
	0x0000 03E8 is 1000 as Int32

address '/**foo**' with an int (1001) and float (440.0)	**Notes**

`0x2F (/) 0x66 (f) 0x6f (o) 0x6F (o)`
`0x00 ( ) 0x2C (,) 0x69 (i) 0x66 (f)`
`0x00 ( ) 0x00 ( ) 0x03 ( ) 0xE9 (è)`
`0x43 (C) 0xDC (Ü) 0x00 ( ) 0x00 ( )`
	0x69 'i' and 0x66 'f' for types
	0x0000 03E9 is 1001 as Int32
	0x43dc 0000 is 440.0 as a float

Table 2.3 Examples of two simple OSC messages.

Each OSC message starts with a string containing its "address", encoded via American Standard Code for Information Interchange (ASCII) characters (1 byte each) and terminated by a null character. All data frames are encapsulated using 4-byte boundaries. This

means that anything less than 4 bytes will be zero padded to the full length. This flexibility of representation, afforded by the higher speeds of Ethernet and subsequent Wi-Fi enabled devices have made OSC relatively popular in the DMI community. More advanced utilization of the format include the implementation of recursive hierarchies and wildcards within the address parameter, and the encapsulation of groups of values through "bundles" (Freed 1997), and the main distinguishing feature of OSC, as its name implies, is the openness and relative lack of constraints of the standard compared to MIDI.

However, the consequence of this flexibility is that while it is relatively trivial to build a physically compatible OSC interface using any device that supports the networking stack, there is no standardized way to respond to OSC messages in a normalized way as in the case of MIDI. Indeed, "bridges" that implement specific mappings between system-specific OSC messages and standard MIDI devices often needs to be built to interact with multimedia software or commercially produced devices such as controllers and synthesizers.

2.2.3 Examples of Wireless Sensor Interfaces

Having defined and exemplified a number of possible messaging protocols and physical interfaces that a sensor interface could employ, in this section we present some examples of existing sensor interfaces in the literature as well as some that have been developed into commercial products.

The WiSe Box (Fléty 2005) and x-OSC (Madgewick and Mitchell 2013) are two examples of Wi-Fi 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.

The I-CubeX Wi-microDig is a Bluetooth 2.0 based sensor interface that transmits MIDI formatted messages over a Bluetooth 2.0 Serial Port Profile as it was the only feasible

method of transmitting data over a Bluetooth link when the Wi-microDig was developed (late 2000's). A custom MIDI mapper running on the host computer bridged the virtual serial port created by the Bluetooth link to emulate a virtual MIDI port.

Name	Physical Interface	Communication Protocol
x-OSC	Wi-Fi	OSC (direct)
WiSe Box	Wi-Fi	OSC (direct)
Kroonde Gamma	Proprietary ISM	OSC (ethernet via adapter)
I-CubeX Wi-MicroDig	Bluetooth 2.0	Custom MIDI mapper
Sense/Stage	Zigbee	Serial Line Internet Protocol (SLIP) (via USB-serial adapter)

Table 2.4 Examples of wireless sensor interfaces.

The Kroonde Gamma is a wireless sensor interface where the sensor communicates via a proprietary protocol on the ISM band with the base box that transmits OSC messages via a wired Ethernet interface or MIDI output port (Kitchen 2016). The wireless latency is specified as 1ms while the total UDP transmission is as low as 5ms for OSC. However, the 5ms value did not appear to come from actual measurements, but rather a computation based on the 200 Hz sampling and transmission rate (Coduys, Henry, and Cont 2004). The configuration of this interface is interesting however, in that it breaks down the sensor interface into two separate modules and relies on a wired interface connected to the host receiving device. While the RF protocol for the Gamma was not specified, a number of other sensor interfaces make use of similar connection topologies, such as the ZigBee based system presented by (Fléty and Maestracci 2011). Here, two matching microcontroller based units are used on the sensor interface and host ends to provide wireless connection, and the receiver interface is connected via wired Ethernet and transmits OSC messages.

Table 2.4 presents a list of the aforementioned sensor interfaces along with their physical interface and communication protocol.

Summary

It should also be noted that, at the time of writing of this thesis, only the X-OSC and the I-CubeX companies are available in providing the interfaces. It is pertinent to note that many "emerging" interfaces as described by (Baalman et al. 2010) such as the μParts, EcoMote, and Tyndall Motes have never actually reached the market, or have since been discontinued. This perhaps speak to the powerful forces behind planned obsolescence (Bulow 1986) as well as the general challenge facing the longevity of DMIs (Morreale et al. 2017). On the other hand, increasingly accessible wireless microcontroller and embedded Linux platforms such as the Raspberry Pi (Severance 2013) provide increasingly accessible platforms for implementing sensor interfaces. Regardless of a commercially available solution is used or not, many novel DMIs make use of custom built sensor interfaces as part of the implementation, and it is often not the focus of the work being documented to provide performance evaluations or facilitate reproduction. In such cases the overall description of the sensor interface configuration along with the application itself is often the main way of discerning whether the sensor interface may be adequate for a new context.

2.3 Latency Measurement in DMIs

Since an in-depth analysis of latency often requires breaking down the entire transmission system, it is often not feasible for end-users or application developers due to closed proprietary systems, or the extensive profiling tools required. However, since the measurement of latency involves the recording of the initial time a signal is presented to a system, and the elapsed time until an output is produced, there are empirical methods using relatively accessible audio and general purpose computing tools. These can work especially well for comparative situations where it is more important to obtain the *relative* difference between

systems or components that can be swapped out and compared against each other. In this section we present three general approaches for measuring latency and they include the "Stereo Tape Recorder", roundtrip delay, and end-to-end measurement systems.

Stereo Tape Recorder

The "Stereo Tape Recorder" method is one of the earliest and most commonly employed method for determining the overall latency of a system. This method exploits the high sampling rates and timing accuracy of commercially available digital audio recording equipment. Since audio recording equipment must be performed with strict timing requirements across both channels, any small amount of latency present in the capture process will also be synchronized between the two channels so that the relative timing difference between the two channels can be observed. Figure 2.2 presents a block diagram of this technique.

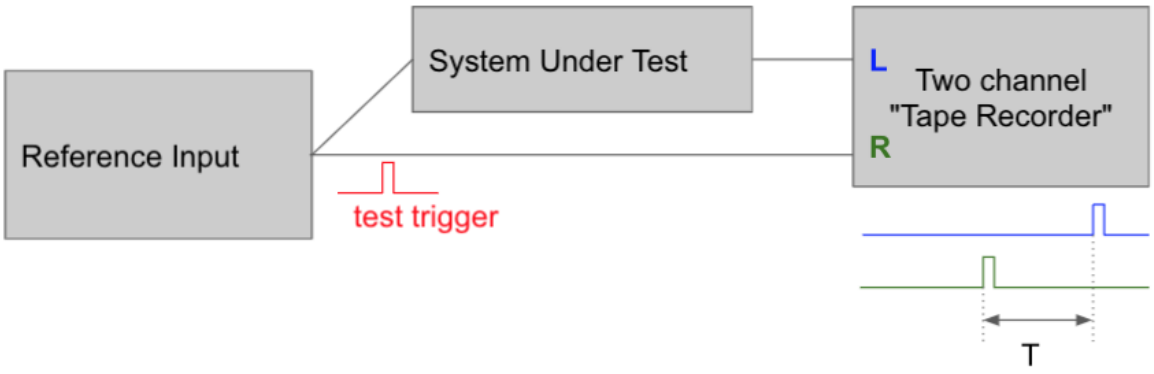


Fig. 2.2 The "Stereo Tape Recorder" model for measuring the timing difference T between two signals.

Equation 2.1 shows, as an example, the measurement resolution using a 44 100 Hz digital recording system, and results in a timing resolution of 22 µs microseconds.

$$\begin{aligned} T_s &= \frac{1}{F_s} \\ &= \frac{1}{44\,100\,\text{Hz}} \\ &= 0.022\,\text{ms} \end{aligned} \tag{2.1}$$

One of the earliest documentation of this process was presented in (Kostek and Czyżewski 1993) where a Digital Audio Tape (DAT) recorder was used to concurrently record the electrical signal associated with the actuation of a pipe organ along with the produced sound captured via a microphone to characterize the timing difference between the two signals. A later analysis by (MacMillan, Droettboom, and Fujinaga 2001) compared the latencies under various conditions and a number of available operating systems at the time, and the results ranged from 2.74 ms in the best case scenario (Linux, unloaded) to hundreds of milliseconds using less optimal driver configurations. Overall, Linux performed the best followed by MacOS, and Windows. The choice of drivers also played a crucial role for all operating systems, favoring Advanced Linux Sound Architecture (ALSA) and Audio Stream Input/Output (ASIO) for Linux and Windows, respectively, while CoreAudio, the only available and universal choice for MacOS provided the best default configuration. The author concludes that the Mac operating system appears to be geared towards audio processing out of the box, compared to the other operating systems.

Beyond the performance of the audio system, it is of interest in the context of DMIs to measure the system latency involving control messages via protocols such as MIDI. Building upon the "Stereo Tape Recorder" paradigm, a translation device, the *Event Transducer* was presented in (Freed, Chaudhary, and Davila 1997) that converted a MIDI or similar digital signal on a wire into an audio-level pulse so that a messages carried via MIDI can be

recorded and subjected to the same analysis based on timing differences between recorded audio channels. While the particular publication did not present any results of its use, this technique was applied by (Nelson and Thom 2004) to measuring the timing accuracy of computer MIDI interfaces where the difference between the raw input MIDI message, and when the operating system has received and re-transmitted the message via the output (MIDI-thru). The measurements revealed that a SBLive on a Windows XP computer (perhaps surprisingly) exhibited the best MIDI performance of around 1 ms average and relatively low jitter. However, one should note here that this is simply the measurement of the MIDI interface performance, and does not provide any indication of audio synthesis performance. Figure 2.3 shows the measurement system, where the measurement system contains both the "tape recorder" functionality via its on-board sound card, as well as providing the test MIDI output triggers.

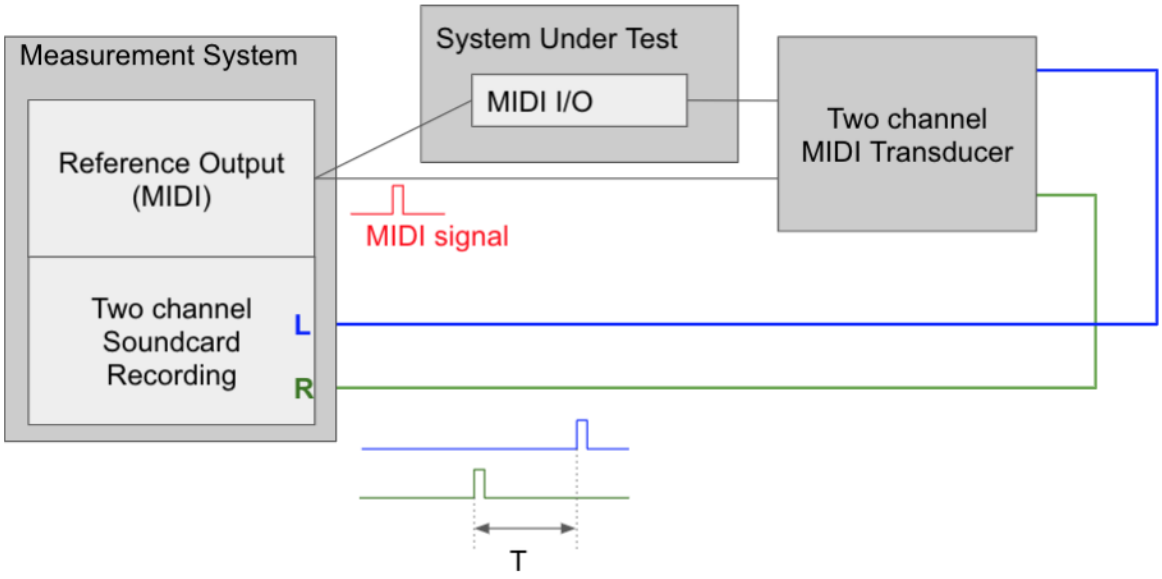


Fig. 2.3 The MIDI performance measurement system by (Nelson and Thom 2004) measuring the timing difference T between a MIDI signal and the same signal passed through a computer's MIDI I/O.

A series of tests were presented by (M. Wright, Cassidy, and Zbyszy 2004) that characterized system audio latency as well as "gesture-to-audio" latency. The system audio latency employed the same "tape recorder" measurements that transmitted audio directly and through the computer's processing system to obtain the audio latency. To obtain the "gesture-to-audio" latency measurements, a microphone was used to record the physical sound of a key press on the laptop keyboard that was then used to trigger the generation of a sine wave in Max/MSP. By evaluating the timing difference between the physical sound emitted when the physical key was struck and the subsequent synthesized sound, the total system delay of this keyboard driven system could be characterized.

One particular challenge introduced by the basic "Stereo Tape Recorder" method, is that for repeated test triggers to be made over time, potentially large audio files need to be recorded and analyzed[1]. A solution to address this was implemented in by adding a real-time analysis component that measured and recorded the histogram of values on the fly in a live buffer (Nelson and Thom 2004), instead of storing the entire audio data itself over time for later analysis, which can result in very large audio recordings.

A variation of this technique makes use of a two-channel signal analysis tool such as an oscilloscope, and (Fléty and Maestracci 2011) employ this method to evaluate a ZigBee-based sensor interface. One limitation of this technique is that it is not as convenient to automate repeated measurements without recording and analyzing the waveforms, and the cost of such specialized analysis tools is also a potential challenge as noted by (Freed, Chaudhary, and Davila 1997). It is also interesting to note that this general concept can even be applied to other mediums where both the source and the resultant output can be captured concurrently, such as the use of a 1000 fps capture to provide 1ms resolution latency measurements of a video system (Wu 2011).

[1] In fact, had the recordings been made on a real tape recorder, the analysis would be even more difficult!

Roundtrip Latency

Using the idea similar to the ping/echo response for network testing, the roundtrip latency of a messaging system can be measured using a matching sender/receiver pair where the receiver will listen for and immediately transmit back a reply as soon as a message is received. The sender records the time the initial message is transmitted, and then measures the amount of time until the echo response is received. Figure 2.4 shows the block diagram of a roundtrip measurement.

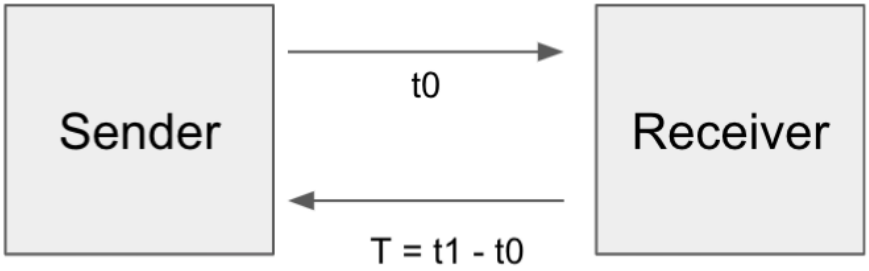


Fig. 2.4 Block diagram of a roundtrip latency measurement.

A performance evaluation of a commercial Wi-Fi OSC interface measured the effect of device scaling and overall bandwidth on roundtrip latency of the X-OSC interface (Mitchell et al. 2014), resulting in a latency of 5.3 ms and 8.09 ms under ideal and loaded conditions. The roundtrip message passing of MIDI messages over BLE between two Linux computers was presented by (Ljungström and Panikian 2016), and revealed that a roundtrip latency of under 10ms was possible for configurations where the connection interval and latency parameters were set to the minimum possible values.

The roundtrip latency is a good technique for measuring the messaging protocol if bi-directional communication is the expected mode of operation, and the uni-directional latency can be considered as half if the system is symmetrical. However, in the case of sensor

interfaces, this is often not the case as the sensor interface may be transmission-only devices that are unable to receive data, or may not process messages at the same rate due to resource constraints. For example, it would be impossible to measure the latency of a keyboard MIDI controller via this method. Also, in the context of DMIs it is often of more significance to measure the end-to-end latency which governs the overall responsiveness of the instrument. Nevertheless, assuming the symmetrical sender/receiver can be configured, the roundtrip latency measurement is a potentially useful tool for isolating the communication protocol alone, and especially useful if the sensor interface may have output/actuation capabilities, such as the case of the X-OSC interface (Madgewick and Mitchell 2013).

End-to-End Measurement Jigs

More recently, an end-to-end latency test system was used to measure the total event to audio output system latency of a number of sensor interfaces and synthesis platforms (McPherson, Jack, and Moro 2016). In this system a microcontroller-based hardware jig emitted a test trigger signal that results in a sensor interface to transmit the message to the host synthesis platform that responds to the trigger message by emitting an audio trigger. The test jig starts a microsecond-accurate timer that then measures the time until an audio output is emitted by the system. Figure 2.5 shows a block diagram of this end-to-end measurement configuration, where the sensor interface and connected synthesis device are contained within the "system under test" block. In some ways, the signal flow from the perspective of the measurement jig is similar to the roundtrip measurement scheme described in the previous section.

Since the system used for the remainder of this work is based on the end-to-end configuration, the implementation will be presented with further detail in Chapter 3.

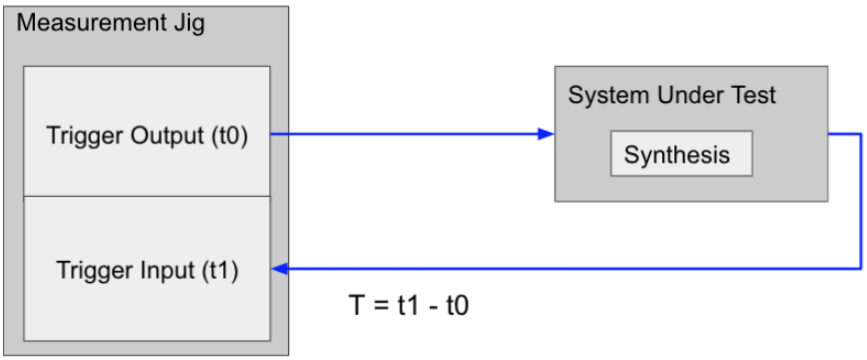


Fig. 2.5 Block diagram of an end-to-end latency measurement system.

2.4 Summary

Having presented an overview of existing sensor interfaces and evaluation techniques that were employed in the context of DMI design, we can now revisit the questions posed at the end of the introductory chapter with a focus on latency as the primary metric of interest. Following are the questions presented in the introductory chapter, along with a description of the method in which we will attempt to address them.

> *What are the minimal possible latencies provided by a particular wireless interface?*

In theory the bandwidth of the physical interface will determine its lowest possible latency, but in practice the protocol (combination of the physical interface and messaging protocol) and implementation will affect its real-life performance. By employing a system that measures end-to-end latency performance along with prototype implementations of particular sensor interfaces for each protocol, we can implement and test the minimum latency performance of each wireless interface configuration that are representative of a real DMI settings.

How fast/often can a particular wireless interface transmit?

As each protocol has specific and limited bandwidth available, this value will affect how fast and how often an interface can transmit. While the transmission rates may be specified by a protocol's specifications, it would be more realistic to provide actual measurements to identify the actual limits under practical situations. By implementing sensor interfaces that can vary the transmission rates, we can empirically establish the point at which the physical limit of the communication channel is reached, resulting in the measured latency exceeding a certain value or messages are dropped completely.

How many wireless interfaces can operate for a particular network configuration?

Similar to the transmission rate scaling situation above, we can increase the number of concurrently operating devices until the latency performance or transmission reliability drops beyond a certain point.

What is the most suitable protocol for a given DMI configuration?

This more general question will involve an integrated view of the findings above, in addition to practical implementation details such as compatibility and interoperability issues that may be relevant in a DMI design context.

Overall, in the light of existing work presented in this chapter on wireless protocols, sensor interfaces, and various measurement techniques used to quantify the performance of various aspects of DMIs, our proposed work will provide an integration of this knowledge into specific conclusions that can guide the design and selection of protocols for DMI design. Additionally, the overall evaluation methodology implemented and utilized by this work can then be applied to cover additional protocols or metrics not presented in this particular thesis.

Chapter 3

Latency Measurement System

This chapter presents the latency measurement system used for the subsequent evaluations presented in the thesis, including a description of the hardware and software implementation of the latency testing jig. Additionally, the characteristics of the signals expected to be transmitted by the tested interface is described to cover the representative situations of trigger and streaming based sampling and transmission schemes.

The test system is built upon the configuration described in previous work (McPherson, Jack, and Moro 2016) with various modifications and improvements in terms of usability and scalability through hardware and wiring interfaces that make it easier to reproduce. The test system consists of a hardware test jig that emits input triggers to a system under test that attempts to synthesize an audio output response, and the time delay between input and output are recorded by the jig. In a DMI consisting of a sensor interface connected to a host synthesis computer, this process reveals the minimal end-to-end latency of the entire system from the moment an electrical input is measured by the sensor interface to the final production of audio by the DAC of the sound card of the system. Functional blocks of each

sub-process in the system is shown in Figure 3.1.

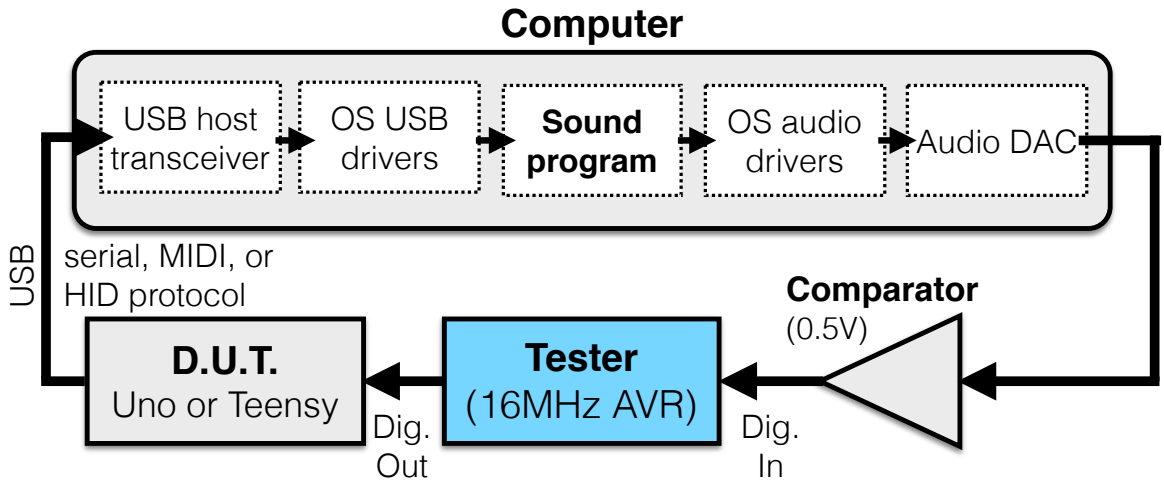


Fig. 3.1 Operational diagram of latency measurement system from (McPherson, Jack, and Moro 2016) (Used with permission).

One key limitation of this procedure is that it measures the total system latency as opposed to the latency of the sensor interface itself. However, if the remainder of the system is kept identical between tests, the differences in measurement then be can attributed to the interface itself. Therefore, in the context of comparing the relative performance of different protocols, this approach can be used to isolate the impact of each protocol. Similarly, this strategy can also be employed to measure different synthesis applications, processing algorithms, or even system audio configurations by changing the variable of interest while holding the others constant.

The measurement jig as presented in the previous work operates in a continuous loop with a fixed 250 ms delay after the completion of each test cycle, henceforth referred to as the *trigger delay interval*. This allows multiple successive measurements to be initiated and logged automatically for further analysis. Figure 3.2 shows the flow diagram of the test jig. In the original work the experiments were performed using 1000 measurements for each device, or around 250 seconds of data assuming the 250 ms (4 Hz) delay interval between

measurements. However, the actual total duration for the total test is slightly longer since each complete cycle duration is made up of the sum of the *trigger delay interval* and the actual measured latency response of the system as well, which is typically up to a few tens of milliseconds based on the presented results (McPherson, Jack, and Moro 2016). For example, if the average measured latency of the system was 50 ms, then the total loop time for each measurement would be 300 ms, leading to about 300 seconds for the 1000 measurements.

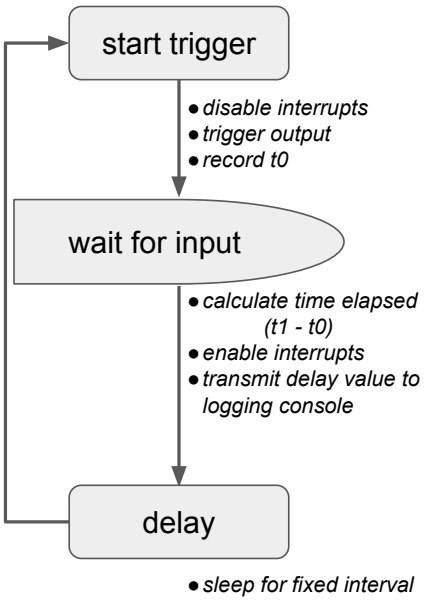


Fig. 3.2 Flow diagram of test jig.

3.1 Hardware

The test jig hardware, as depicted by the "Tester" in Fig 3.1, is a microcontroller that emits a test trigger signal that goes through the system under test, and measures the duration until a response is received. The measured delay value is sent back to a host terminal for logging purposes via a serial port attached to a USB serial adaptor built into

the development board. The particular implementation makes use of an Arduino Uno board, which contains a 16 MHz AVR ATmega328 as the central processor and is one of the most widely used boards within the Arduino (Kushner 2011) Ecosystem. While this board is relatively simple and lacks additional features like wireless capability, faster/multi core processing, or built-in peripherals like battery charging, its simplicity, low cost, and large user base make it a very accessible platform. Additionally, for the kind of latency measurements in question, the main requirement is to have sufficiently high resolution and deterministic timings, rather than processing power or other peripheral features.

As described in previous literature, the timing critical section of the measurement phase performs a series of functions to disable external interrupts and other firmware features so that the time difference being measured is deterministic and clock-accurate at 1/16 µs intervals for a 16 MHz clock (McPherson, Jack, and Moro 2016). Therefore, we can proceed with any subsequent analysis that the system will provide relevant measurement resolutions based upon this value. Considering that most of the relevant measurements surrounding musical interactions are typically performed at the millisecond scale (Rottondi et al. 2016), (Jack et al. 2018), this three orders of magnitude resolution is sufficient for our application.

While no specific instruction was provided in the original literature describing this hardware (McPherson, Jack, and Moro 2016), we were able to reproduce it based on the description from the original publication. The main hardware addition of the latency measurement jig, beyond the microcontroller board itself, is a comparator circuit that translates the resultant audio output signal into a digital trigger for the return signal timing measurement. This circuit effectively allows the conversion of the synthesized analog audio output into a trigger for the accurate timing detection of the final system output. The first version of the hardware was prototyped on a through-hole perfboard, as shown in Figure 3.3. The core of the comparator hardware is implemented using the LM393 IC (Texas

Instruments 2020). Once the prototype was verified to operate as intended, a more robust and modular PCB version was designed to aid in the usability and reproducibility of the test system.

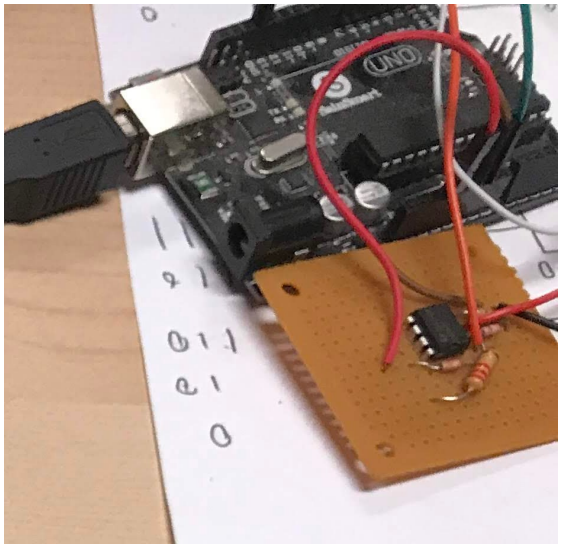


Fig. 3.3 The comparator circuit implemented on a protoboard.

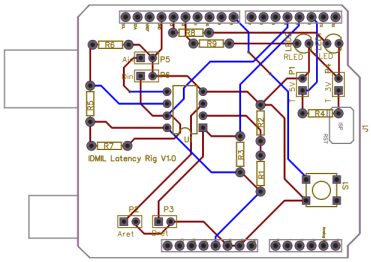


Fig. 3.4 PCB layout of latency rig.

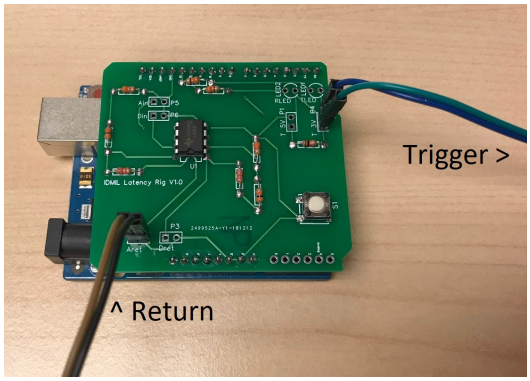


Fig. 3.5 The assembled latency jig PCB installed on an Arduino Uno.

An Arduino Uno-compatible "Shield"[1], depicted in Figure 3.4 was created to house the additional circuitry including the comparator IC as well as supporting headers for

[1]A common term used in the Arduino community to describe add-on boards that have the same form factor and pinout as the base Arduino board in question

connections from the resultant audio output jack and to the trigger input of the device under test.

The shield design allows easy construction, installation, and removal from the main development board for modularity as it matches the shape and pinout of the base board. Through hole components for the entire design was selected as there was plenty of area for placement, and while they are marginally more expensive, the cost is negligible and makes the assembly process considerably easier than surface mount components, which further encourages reproducibility when miniaturization is not required (Franco and Wanderley 2016a). The design files and bill of materials have been archived on public Github repositories so that they can be easily modified, or sent to PCB manufacturing services for production, and are also included in Appendix B.

3.2 Software

We employ the same software environment for receiving and synthesizing audio that was implemented in the Max/MSP framework, using the same synthesis components (McPherson, Jack, and Moro 2016). The synthesizer consists of a simple patch that receives the incoming trigger from the sensor interface under test, and immediately emits a decreasing ramp with a period of 50 ms. Figure 3.6 shows an example of the patch, with the input being triggered by a *notein* object for the case of MIDI input triggers. For tests involving OSC, a *udpreceive* object is used instead. The audio output is sent to the return input on the latency measurement jig via the system's sound card.

The computer used for the majority of tests described in this thesis was a 2015 MacBook Pro with a 2.5 GHz i7 processor, 16 GB Random Access Memory (RAM) running Max/MSP 7.3.6 using the on-board audio at 44.1 kHz, 32 I/O and signal vector sizes and

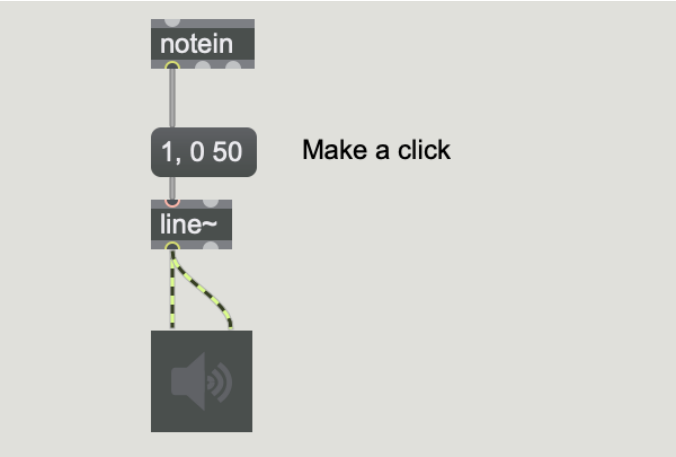


Fig. 3.6 The basic synthesizer patch that creates a decreasing ramp of 50 ms

"scheduler in overdrive", consistent with the configuration used in for best latency performance (McPherson, Jack, and Moro 2016). While certain specialized configurations, as described in (Y. Wang 2018) could potentially yield better audio performance from a host system perspective, in our comparative analysis it is more important to keep the system constant as well as representative of potential user application settings (which suggests a commodity off the shelf system, such as a Mac workstation). It should be noted here that, with a I/O and signal vector sizes of 32, the duration of each audio block that is discretely processed by the system and transmitted to the sound card is the duration of the 32 samples at 44.1k samples per second, or 0.73 ms.

The duration of this block will define the minimum timing resolution that the system is capable of achieving, as an event that arrives between buffers can only be represented in a subsequent block. Additionally, some audio systems may include multiple buffers, which will then add to the overall latency of the system. However, in the situation of making relative comparisons, as long as this setting is unchanged between measurements, this value will be constant.

3.3 Sensor Interface and Test Signal Characteristics

It is difficult to describe the performance of sensor interfaces without some understanding of the nature of the signals of interest. In this section, we describe some basic characteristics of the signals that the sensor interface will capture and transmit, along with how the test jig will operate in relation to it.

3.3.1 Sensor Interface Signal Characteristics

While sensors themselves may require processing time depending on their characteristics, the minimal delay of a digital input is a good way to isolate the latency introduced by the sensor interface. Any sensor that may require additional pre-processing time to would of course add to the overall latency. An example could be a peak detection algorithm (Palshikar et al. 2009) that takes a continuous signal and generates a single trigger event when a peak in the signal occurs. In this case, the detection of the trigger will likely require additional samples to be captured after the peak to accurately identify it. In our evaluations, we focus on the effects due to the sensor interface (rather than the sensor themselves) and as such, do not measure the sensor processing or sampling time that may be present for physical signals and specific to particular sensors. In real life these delays will of course depend on the physical characteristics of the sensor itself as well as any processing of the signal that might be required.

What is relevant, however, is how a sensor interface samples and transmits the messages. As mentioned in Chapter 2.1, the interface may operate in a trigger/event-based, or continuous streaming mode. The first can be used for sporadic, discrete events such as note on/offs, or toggles and buttons on a MIDI controller. In such cases, an interrupt-driven system can be used to trigger the transmission of a signal as soon as it is received,

and the main delay of this process will be attributed to the transmission delay alone. For continuously varying data such as knobs and sliders or streaming sensor data from Inertial Measurement Units (IMUs), it is more suitable to constantly sample and transmit the data at a rate that captures the relevant changes in the signal. In this situation, since the sampling must be done at finite intervals, this process will add a delay depending on this sampling interval. This exact frequency would depend on the characteristics of a particular signal that needs to be captured, and the sampling theorem dictates that double the maximum frequency of interest be used. These two transmission schemes are depicted in Figure 3.7. If we assume the sensor to have negligible processing delay for the purpose of our analysis, then the trigger-based sampling and transmission scheme will add no additional latency while the streaming configuration will, on average, exhibit an additional delay of half the sampling period. This consideration is important as for an example sampling rate of 100 Hz ($T_l = 10$ ms), the additional latency due to sampling is 5 ms, and exceeds the *entire end-to-end latency* of the fastest wired interfaces of 5.1 ms in (McPherson, Jack, and Moro 2016).

The other consequence of the streaming transmission is that continuously sampled data must be transmitted by the sensor interface, regardless of whether specific triggers or events occur. This leads to a larger amount of data and can have an impact on the scaling behaviour as the bandwidth of the transmission channel is saturated, and is a metric of interest for our subsequent evaluations.

3.3.2 Test Signal Characteristics

The input provided by the measurement jig is step-wise signal that is nominally low and switches high to indicate the start of the test trigger, and held until the jig receives the produced audio output from the system. In real life settings such an event would warrant a

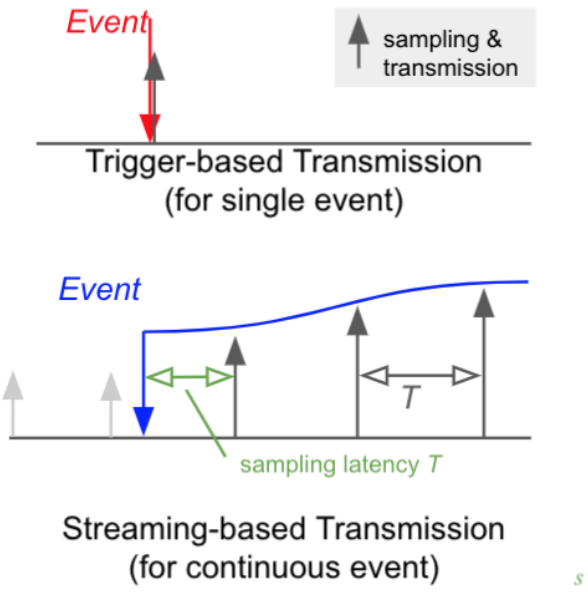


Fig. 3.7 Trigger and streaming transmission.

trigger-based message transmission strategy, but since we would like to test the behaviour of the sensor interface under both continuously streaming situation as well, we can also repeatedly sample and transmit the input value for the streaming transmission mode. In the latter situation, the system will be constantly receiving the same input signal, but the synthesized result will only be generated upon reception of the trigger event. The test is repeated for multiple measurements over time where the delay is much longer than the duration of the synthesized audio. In the work described in subsequent chapters, we employ the trigger or streaming version of the test depending on the needs of the specific evaluation. In general, when attempting to determine the minimum latency of the system, the trigger-based transmission scheme is employed while the streaming-based scheme provides an idea of the maximum real-time sampling rate that can be used.

3.3.3 Model of Latency Behaviour and Test Triggers

Based on the block diagram from (McPherson, Jack, and Moro 2016) in Figure 3.1, we break down our test system in a similar way when considering the measurements relating to the wireless protocols being tested in this work. Since many processes occur at extremely small time intervals on the host computer, especially when considering the microsecond resolution of the latency measurement jig, it is not feasible nor relevant in this analysis to "zoom in" at such levels of detail. Instead, in Figure 3.8 we present the components of interest that are directly associated with each wireless protocols of interest.

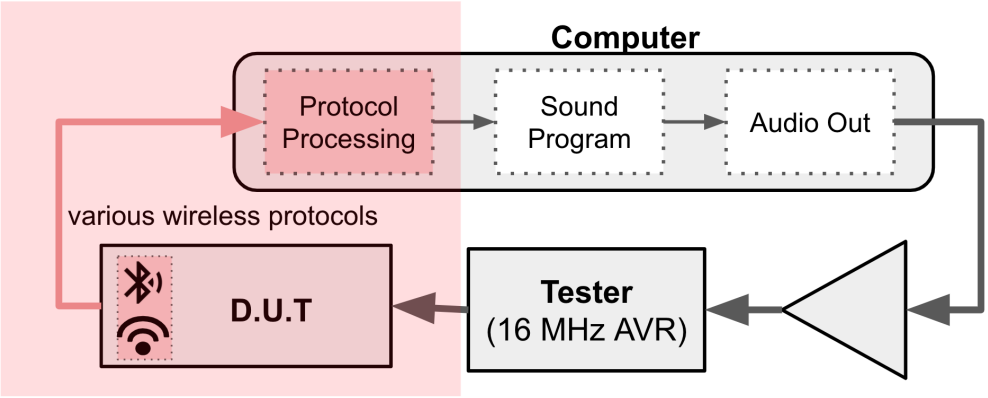


Fig. 3.8 Block diagram of measurement system highlighting the components under test.

The overall latency can be expressed as $L_T = \sum_{n=1}^{N} L_n$ where L_n is the latency of each sub block/component in the system. In general, the expected behaviour of each component would follow a distribution depending on the underlying mechanisms involved.

For example, the synthesis system is subject to the computer operating system's context switching, but the audio output has very distinct buffers that are processed at relatively large block sizes. Empirically, given the microsecond resolution of the latency jig described previously, there are three main discernible processing intervals due to the following:

- **Wireless Interface:** This is the primary metric of interest, and internally can be relatively small in the case of Wi-Fi (sub-millisecond transmission windows), or larger in the case of the connection interval for Bluetooth (minimum 7.5 ms).

- **Audio Block Size:** Most sound cards on general purpose operating systems transfer an entire block of samples at a time, and the size of this block has a direct consequence in how quickly. This is referred to as the "I/O Vector" size in Max/MSP. The specialized embedded Linux environment (McPherson and Zappi 2015), provides the possibility for single sample delays between sensor input and audio output.

- **Signal Vector Size:** In environments such as Max/MSP, this controls how many samples are computed by the synthesis environment at a time.

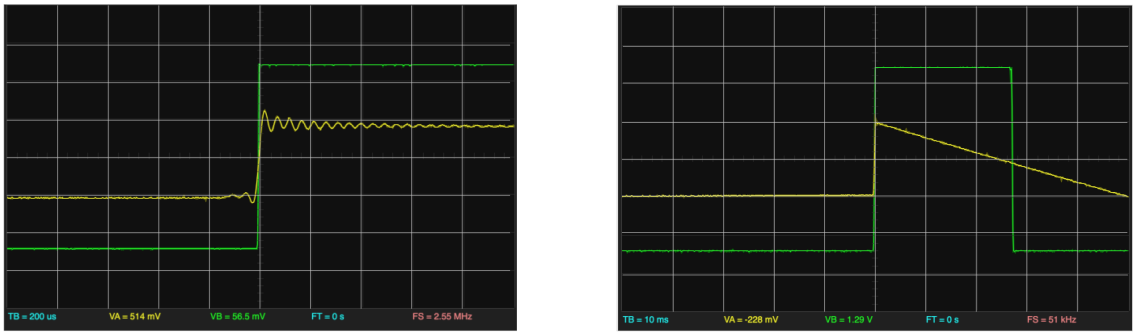
The latency introduced by the wireless interface will depend on a number of factors, but the most basic parameters consist of how much bandwidth the channel has, and the length of the message. It logically follows that the larger the message, the longer it would take to transmit over the air for a given data rate. As a rough approximation, the time (in seconds) that a message requires for transmission is given by $\frac{M}{B}$ where M is the number of bits in the message, and B is the bandwidth (in bits per second). Of course, the actual protocol, and various supporting layers in the stack as described in Chapter 2.2 will add additional overhead to this minimal value. Additionally, a protocol such as BLE may only transmit within specific, synchronized windows which then impose further limits, as an imminent event may have to wait until the next interval in the same way that sampling intervals may introduce latency as described in Section 3.3. However, since our analysis takes place at the user application level, where we select and compare existing messaging protocols, we are subject to the minimal implementation standards defined by the protocol.

3.4 Preliminary Tests

In this section we describe some preliminary tests that verifies various aspects of the test system. First we evaluated the behaviour of the hardware comparator of the test jig, followed by the use of the system in a basic end-to-end measurement of a wired USB-MIDI interface to compare with previously published results.

3.4.1 Hardware Comparator Operation

To ensure that the produced audio output interacts correctly with the hardware comparator circuit, the audio output and comparator outputs were measured with an oscilloscope. Figure 3.9 presents the scope measurements of the synthesized output ramp single, along with the comparator circuit's output trigger that will be received by the latency measurement jig. The closeup (200 µs horizontal grid size) reading shows the immediacy of the comparator output response associated with the ramp input, while the zoomed out version (10 ms grid size) shows the entire ramp that spans 50 ms.



200 µs grid size 10 ms grid size

Fig. 3.9 Oscilloscope plots showing the emitted audio (yellow) and comparator circuit (green) outputs.

3.4.2 USB-MIDI Test

To ensure that the system produced will generate comparable results from previous studies and verify that we have accurately reproduced the test system, we re-implemented the wired sensor interface consisting of a Teensy microcontroller operating as a wired USB-MIDI interface as a "sanity check". This is one of the fastest tested configurations according to previous literature (J. L. Wright and Brandt 2001; McPherson, Jack, and Moro 2016). Figure 3.10 shows a block diagram of this test setup. The results from this test, using as close to the original specifications as we were able to reproduce, yielded an average latency of 4.1 ms, which is slightly lower than the value of 5.1 ms presented in (McPherson, Jack, and Moro 2016). This difference could be attributed to the slightly newer hardware and software configurations (2015 vs. 2014 MacBook, OSX 10.14 vs. 10.10, Max/MSP 7.x vs. 6.x).

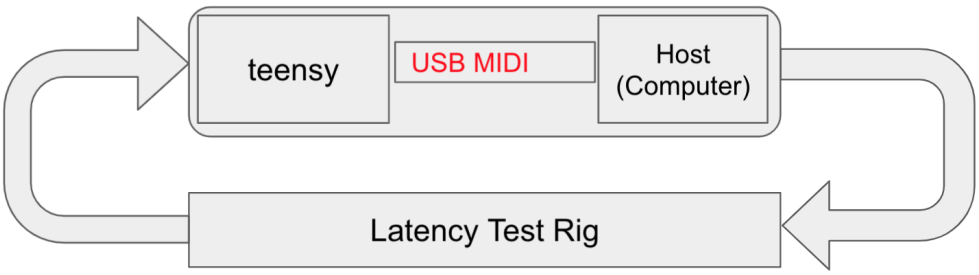


Fig. 3.10 Block diagram of the USB-MIDI test.

3.5 Summary

In this chapter we presented the measurement system in detail, and described the implementation details of the main latency measurement jig used for the remainder of the work.

In the subsequent chapters, we will present a number of evaluations that were performed using variations of this particular configuration.

Chapter 4

Performance Evaluations

In this chapter we present three sets of evaluations of latency from various perspectives. Section 4.1 presents the scaling performance of OSC over Wi-Fi, while Section 4.2 focuses on BLE and comparison with some alternatives from the perspective of designing a new commercial wireless sensor interface. Finally, Section 4.3 performs the comparison of three protocols from the perspective of the Internet of Things. Section 4.4 presents the scaling evaluation of BLE. Combined, they address from different perspectives the questions posed in the introductory chapters of the minimal latencies, maximum transmission rates and number of concurrent devices in operation for a given transmission rate, leading towards the subsequent concluding chapter that presents some more holistic views on wireless protocol choice in the context of DMI design. Finally, in Section 4.5 we present two adaptations of the evaluation system in collaborative works where the performance of embedded Linux systems used to implement DMIs, and a hybrid motion capture system.

4.1 Wi-Fi Scaling

In this section we look at the scaling performance of one wireless sensor protocol: the transmission of OSC packets over Wi-Fi. In particular, the number of devices and send rates were altered while the end-to-end latency of the entire system was measured according to the procedure described in Chapter 3. The findings of this work was published in (J. Wang, Meneses, and Wanderley 2020).

One of the key benefits of wireless sensor technology in the DMI context is that it is possible to create a network of multiple, untethered devices to facilitate ensemble performance settings where the performers are not necessarily constrained to fixed locations. Motivated partially by the increasing number of wireless devices and platforms used at IDMIL, in this work we investigated the scaling behaviour of a commonly used protocol, OSC (M. Wright 2005) implemented on a wireless microcontroller platform, the ESP32. The ESP32 is a system-on-chip microcontroller that provides single and dual core processor configurations running at up to 240 MHz, with an integrated radio that supports both Bluetooth Low Energy (BLE) and Wi-Fi[1]. As a low-cost embedded solution with development boards readily available for around $10, the ESP32 is an ideal solution for the construction of sensor interfaces and controllers that need wireless connectivity, including the GuitarAMI (Meneses, Freire, and Wanderley 2018) and new iterations of the T-Stick[2]. OSC was chosen for this analysis as it is widely used as well in existing projects, and Wi-Fi has considerably more bandwidth than BLE. Besides being one of the first evaluations of this nature for the ESP32, this work is also the first time that such scaling behaviour is evaluated for similar sensor interfaces in the literature.

[1] `https://www.espressif.com/en/products/socs/esp32`

[2] The first wireless implementation of the T-Stick (Nieva et al. 2018) employed the related ESP8266 microcontroller.

The two main metrics of interest in this work include the number of devices, and the transmission rate per device. Under an initial assumption, it would seem reasonable that N devices transmitting at rate M is equivalent to a single device transmitting at $N \times M$ from a bandwidth perspective, but in practice there are overheads associated with switching between each device in through a common shared wireless channel. Therefore, the exact relationship is not as straight forward and one method to obtain the scaling behaviour is through empirical measurements.

4.1.1 Overview

The overall setup of the latency measurement is described in Chapter 3, and Figure 4.1 presents a block diagram of the test setup. A D-Link DIR-601 Router running LEDE 17.01 was used to provide the Wi-Fi access point. The host computer was a quad core i7 2.5Ghz MacBook Pro running OSX 10.14 and Max 7.0.1 was used for the OSC receiving and synthesis environment that received the input triggers and produced the audio output. The OSC message consists of a very short message with an address of '/a', and a single integer value. Due to the 4-byte boundaries and zero padding of the OSC specification (Freed 1997), the size of this message will be the same as the first example in Table 2.3 despite the slightly shorter address, and is 12 bytes in length[3].

Since we are primarily interested in the scaling behaviour with a controlled increase in send rates, but also want to avoid the latency due to the sampling/send rate as described in Chapter 3.3.3, the test firmware was set to continuously send 0's corresponding to a note off at the configured send rate unless the trigger input was detected, in which case a 1 was emitted *immediately*. This method allowed us to test both the minimal transmission latency

[3]In this analysis we look at our application level payload sizes, while acknowledging that the lower levels of the networking stack will add additional headers and termination bytes beyond our control

that is not subject to sampling delays, while at the same time increase the "streaming" rate to observe the effects of network saturation as more devices are transmitting. Figure 4.1 shows the test setup, where the final end-to-end latency is calculated by the time difference between the trigger output and audio input.

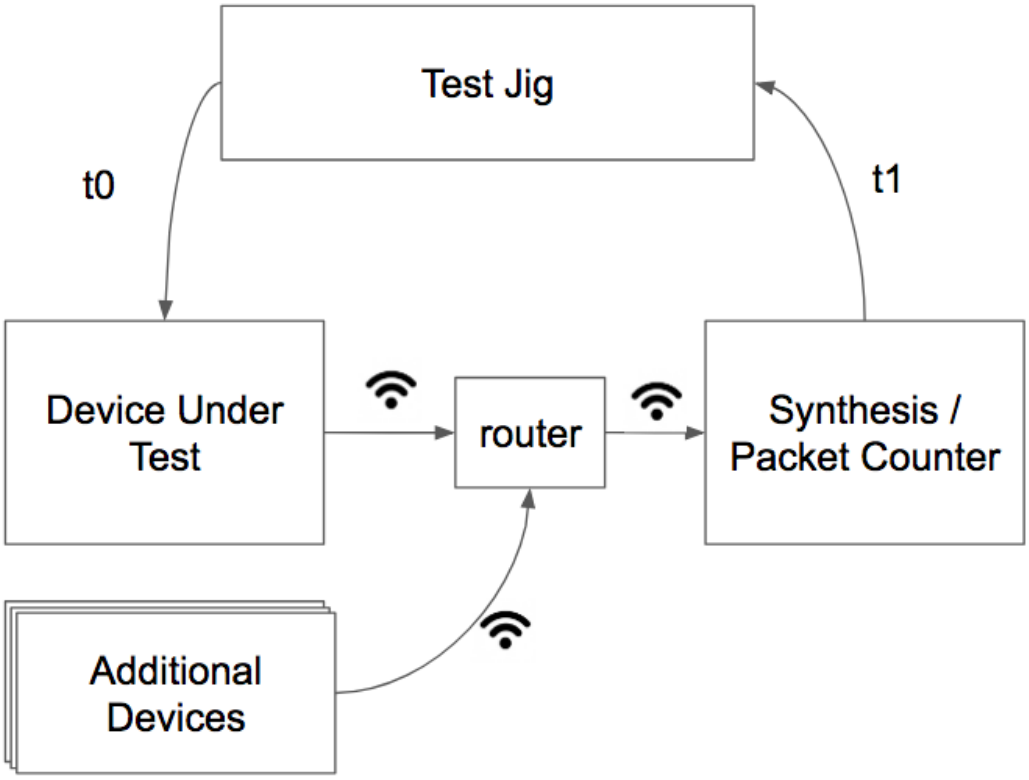


Fig. 4.1 Device and send rate scaling test configuration.

The majority of the tests presented above were performed in an office environment on a weekend, where there was little traffic. A single test was also carried out during a work day to compare the difference in performance when more potential network congestion was encountered.

We employed a wireless access point with no encryption (as suggested for higher performance (Mitchell et al. 2014)), and selected a channel that had the least amount of frequency

usage measured by a MikroTik[4] router. This router provided some basic channel utilization tools (Figure 4.2) that allowed us to select the least busy channel, and we operated this device as a separate scanning tool alongside the system being tested. The tests were performed in an office environment on a weekend where there was relatively little network traffic.

#	Frequency (MHz)	Usage	Noise Floor
0	2412	24.1	-103
1	2417	21.2	-104
2	2422	7.0	-104
3	2427	0.0	-104
4	2432	6.5	-105
5	2437	12.8	-101
6	2442	4.8	-103
7	2447	1.0	-102
8	2452	3.4	-103
9	2457	12.2	-106
10	2462	9.3	-105

Fig. 4.2 Screen capture of the MikroTik's "Frequency Usage" tool, showing activity in each Wi-Fi channel. "Usage" is a measurement in percent, while the Noise Floor is specified in dB.

4.1.2 Single Device Send Rate Scaling

In this test, a single device was used to transmit at increasingly higher intervals. Figure 4.3 shows the measured latency as the transmission rate was increased, up to a maximum limit that was obtained through empirical measurements. The transmission rate was controlled using a loop timer that determined the time elapsed between successful sampling and transmissions and set to operate at the desired rate. The maximum limit of 2300 Hz was

[4] `https://www.mikrotik.com`

obtained when the sensor interface was configured to operate as quickly as possible with no additional delay. From here we see that we are able to achieve below the 10 ms latency value at 1000 Hz. At this rate, the bandwidth is $1000 \times 12 = 12000\, bytes/second$. Despite the generally increasing trend between latency and transmission rates, we notice a slight decrease in measured latency when increasing the transmission rate from 100 to 200 Hz, and this suggests that there is a certain amount of variability between measurements.

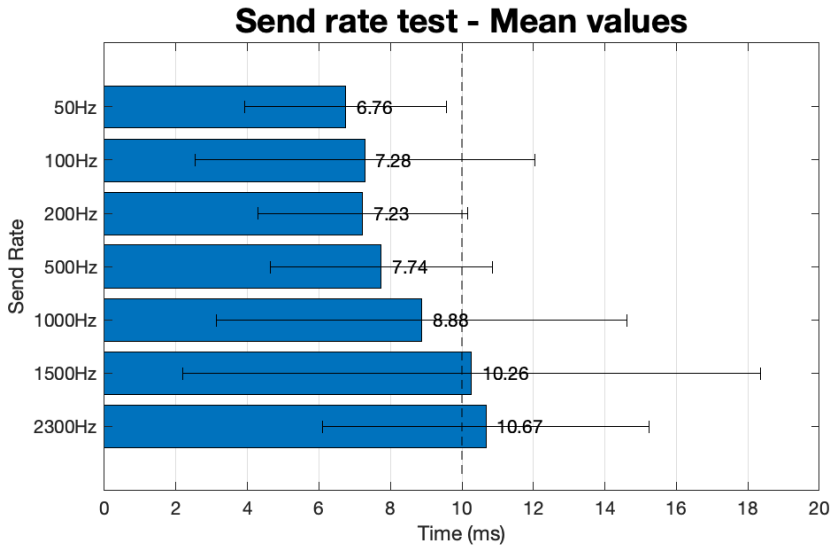


Fig. 4.3 Latency as a function of transmission rate. Error bars indicate standard deviation.

Another presentation of the data that can reveal interesting characteristics of the distribution is the Empirical Cumulative Distribution Function (ECDF), as presented in Figure 4.4. This probabilistic function plots the latency values on the X-axis, and the cumulative probability of that particular latency value on the Y-axis. By looking at the value on the vertical axis for a given threshold (e.g., 10 ms, as represented by the dotted vertical line), one can easily identify the percentage of measurements that meet a particular threshold. For example, at 1000 Hz, or the highest tested transmission rate that had an average of below 10 ms, 75% of the values were within this threshold. At the maximum possible trans-

mission rate of 2300 Hz, only about 55% of the latency values were below 10 ms. Even at the lowest transmission rate of 50 Hz, there were 10% of values above the 10 ms threshold. The other pattern observed as the transmission increased is the reduction of the slope of the ECDF, which signifies a wider variance (jitter) of the measured values.

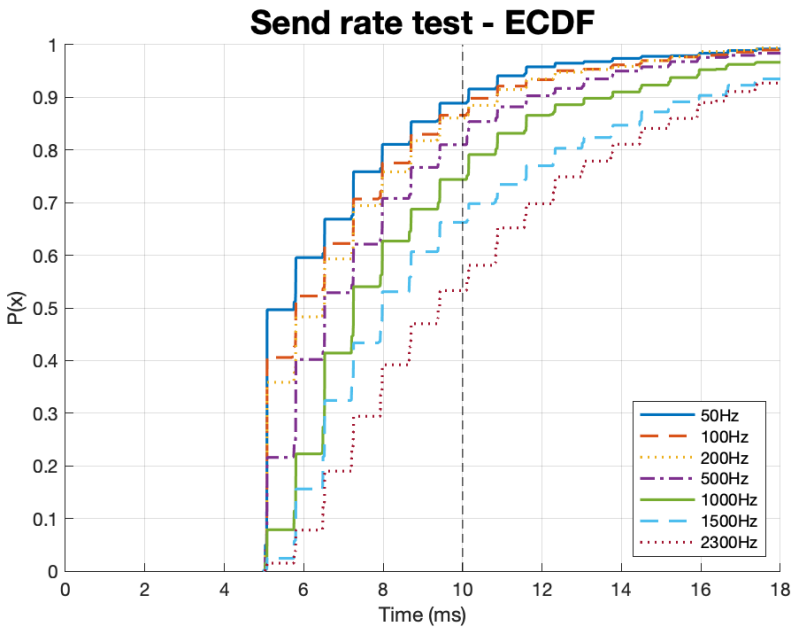


Fig. 4.4 The Empirical Cumulative Distribution Function of latency values for a single transmission device operating at transmission rates from 50 - 2300 Hz.

Given the maximum transmission rate of 2300 Hz and 12 bytes per message, we can then calculate the maximum bandwidth of $2300 \times 12 = 27600\ bytes/s$, or $220800\ bits/s$.

4.1.3 Multiple Device Scaling

We next measured the effect of adding multiple concurrent devices that are transmitting at the same time. Here, while only a single sensor interface was used to measure the latency through the jig, the other devices were set to continuously transmit concurrently without being triggered. Figure 4.5 shows the average measured latency as additional devices were

added, starting from 0 (which is just the single triggered device operating) to a total of 13 devices, which were the maximum number of hardware boards we had available for testing at the time. Here, we see that it was possible to maintain an average of below 10 ms for up to 6 devices. At this point, given the 100 Hz transmission rate, the bandwidth is $6 \times 10 \times 12 = 7200\, bytes/second$, which is considerably lower than the 12000 bytes / second achieved in the single device case.

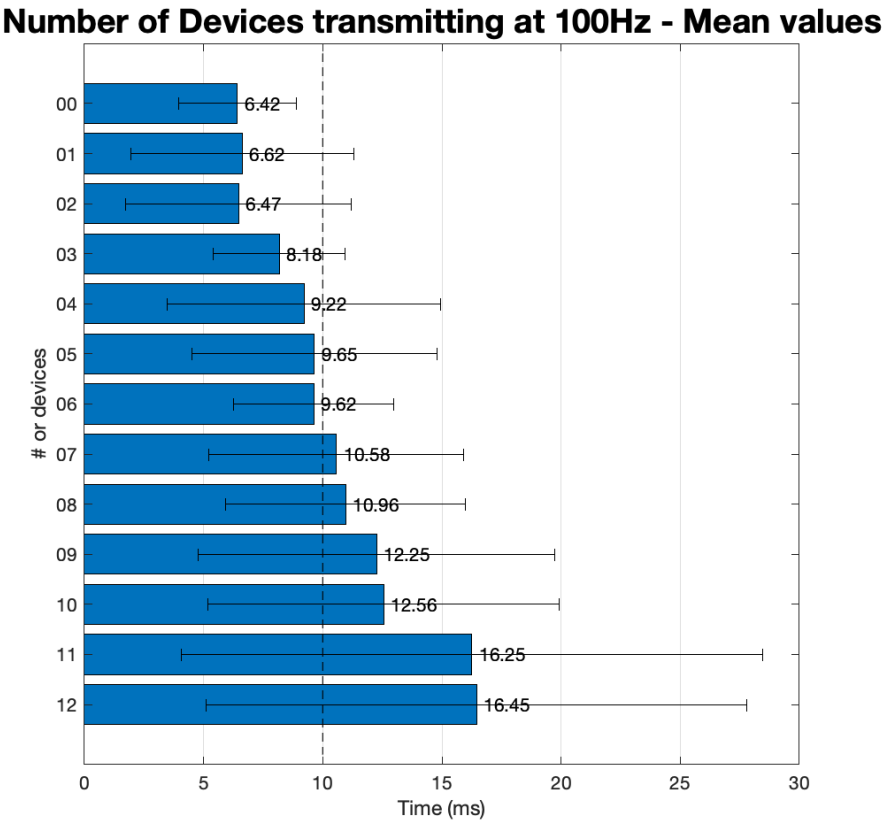


Fig. 4.5 Latency as additional devices are added (0 implies only the single device being measured is present).

The ECDF of the latency values are presented in Figure 4.6, and shows that while for 6 devices the average latency may be within the 10 ms threshold, only around 70% of the

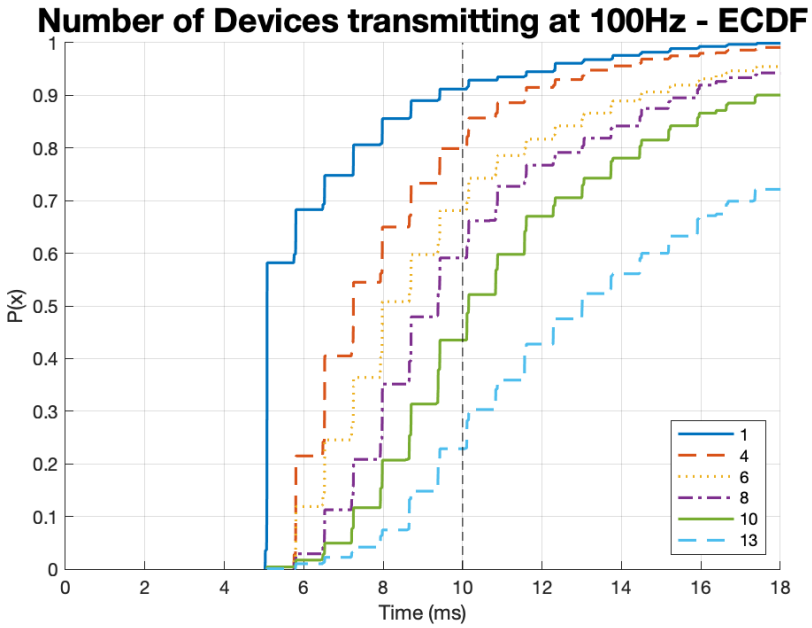


Fig. 4.6 Empirical Cumulative Density Function (ECDF) of the multiple devices latency test.

values are below this value.

4.1.4 Network Environment Effects

Finally, while we attempted to make most of our measurements on an uncongested network by working on a weekend, we also wanted to see if a busier networking environment had any effects on the latency measurements. Figure 4.7 shows the 100 Hz test taken from 25, and 26 January, 2020, as well as the 30th. The results show significant difference between the first two days that were on a weekend and the third, which was a weekday. Even though the router was used exclusively for the experiment and no other wireless devices were active on the local network, this result suggests that background wireless activity from nearby access points on the same channel can have a significant impact on the performance.

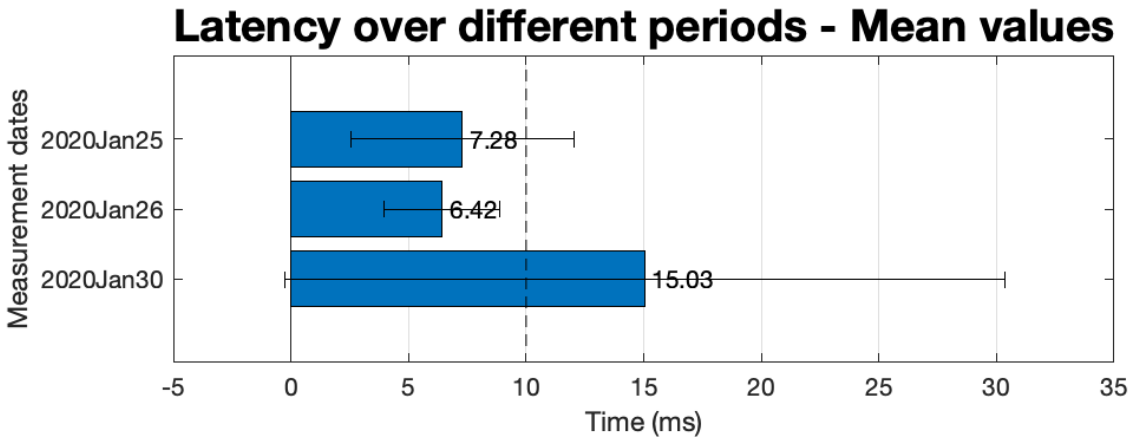


Fig. 4.7 Mean values of the 100 Hz single-device latency test across different days.

4.1.5 Summary

In the Wi-Fi OSC send rate and scaling tests implemented on an ESP32 microcontroller as a sensor interface, we obtained end to end latency measurements for 1 to 13 concurrent devices operating on a single Wi-Fi channel. In general a trend of increasing latency is observed as more devices or higher transmission rates are used, although there is a small amount of

variation between tests as evident by the non-monotonically increasing results presented between 100 and 200 Hz transmission rate in the single device test, increasing between one and two additional devices in the device scaling test, and the between-day measurements. More significantly, the background activity that may be beyond one's control can have a significant impact on the performance of a Wi-Fi system, as evident from the tests performed on different days of the week in an office environment.

4.2 I-CubeX WiDig and BLE Implementation Considerations

4.2.1 Overview

In this section we present the work that was done with Infusion Systems in the investigation and subsequent development of a new Bluetooth-based wireless sensor interface, with a focus on the use of BLE as a wireless interface and its performance and usability characteristics. The main findings were published in a conference paper at NIME 2019 (J. Wang, Mulder, and Wanderley 2019).

The I-CubeX WiDig is a wireless sensor interface that was developed as part of an industrial collaboration with Infusion Systems, creator of the I-CubeX sensor platform (Mulder 1995). As one of the earliest commercially available sensor interface ecosystems that was widely available to the artistic community, the I-CubeX platform has been used by thousands of customers around the world ranging from interactive installations, theatre exhibits, musical performances, and university laboratories. In this section we present an overview of the existing I-CubeX sensor interfaces, the motivation to develop the new WiDig and subsequent evaluations performed.

4.2.2 Previous Interfaces: USB-microDig and Wi-microDig

The sensor interfaces prior to the development of the WiDig were the USB-microDig and the Wi-microDig. Both of these are 8-bit microcontroller based sensor interfaces that support 8 channels of analog or digital i2c sensor input, and up to 8 channels of Pulse Width Modulation (PWM) or digital output (when digital inputs are not used). The USB-microDig (Figure 4.8) interfaced with the computer via a virtual USB serial port implemented with a SiLabs USB-serial IC, while the Wi-microDig employed a Roving Networks Bluetooth 2.0 Serial adapter implementing the Bluetooth 2.0 Serial Port Profile (SPP). Both of these devices appeared to the host operating system as a virtual serial COM port device.

Fig. 4.8 I-CubeX USB-microDig.

Fig. 4.9 I-CubeX Wi-microDig.

One unique feature of the I-CubeX sensor interfaces was their ability to perform a number of configurable mapping tasks. Raw sensor input can be configured to undergo a series of processing include scaling, thresholding, and conversion to MIDI messages which can be sent directly to synthesizers. However, in both cases, due to the virtual COM ports implementation (via USB or Bluetooth) for communication with the host it was necessary to operate a bridging application to implement a virtual MIDI port on the host computer.

In terms of latency performance, prior informal testing at Infusion estimated the USB wired interface capable of sub-10 ms latency, while the Bluetooth Serial version of the Wi-microDig exhibited latencies above 30 ms.

The development of the WiDig attempts to address two objectives highlighted by the long-term plans of Infusion Systems: First, to achieve an under-10 ms end-to-end latency between a wireless sensor input and audio output, and second, to increase the compatibility and usability of the sensor ecosystem. As wireless interfaces became more accessible along with the growth of increasingly powerful mobile devices, sensor interfaces that can connect directly to smartphones, tablets, and other embedded wireless devices without a intermediate host would increase the convenience and usability of the I-CubeX ecosystem. For the WiDig, both MIDI and OSC messaging protocols were of interest since the existing I-CubeX tools and configuration environment made heavy use of MIDI messages, while OSC was an increasingly popular standard.

4.2.3 WiDig Development

After investigating the current sensor solutions, an ESP32-based interface was added to the existing wired USB-microDig interface since it provided USB, Wi-Fi and BLE interfacing options while reducing the amount of modifications required for the core sensor processing hardware and firmware. To further reduce development time, the base USB-microDig PCB was modified to interface with a pre-manufactured ESP32 board, the Lolin D32.

The resulting WiDig design consists of a very similar production process compared to the existing USB-microDig, but with an additional integrated and cheaply available ESP32 add-on board that provides the USB and wireless interfaces supporting both Wi-Fi and BLE. Due to the additional computation and storage capabilities of the ESP32, this platform also allows future expansion in terms of supporting additional sensor processing,

and even self-contained configuration and control applications such as web interfaces that can be completely embedded within the device.

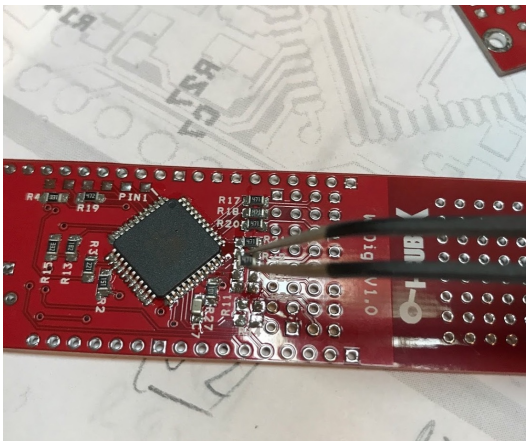


Fig. 4.10 WiDig PCB assembly at IDMIL.

Fig. 4.11 The completed I-CubeX WiDig.

The WiDig was first assembled in small quantities in the lab so the design can be tested before being sent for mass production. Figure 4.10 shows a PCB being assembled at IDMIL, and 4.11 shows the final completed product, with the Lolin D32 board attached on top of the base PCB. Since it was a single layer board, it was possible to perform Surface Mount Technology (SMT) manual assembly by hand and a hotplate reflow process. A batch of 10 boards were assembled and tested this way before larger-scale production commenced in early 2018.

4.2.4 Evaluation of the WiDig

The initial design work involved implementation of test firmware to ascertain the performance and compatibility with I-CubeX's existing suite of interfacing software, and once minimally functional versions were prototyped, a series of performance tests were performed to verify the latency characteristics of the interface. Alongside the performance evaluation, we also investigated the different protocols from a compatibility and usability perspective.

4.2.5 Latency Measurements

Since the ESP32 supports both Wi-Fi and BLE, it was possible to implement a variety of different protocols on the device to reveal any potential differences between them. Figure 4.12 shows the block diagram of the various test configurations. The USB MIDI test, implemented on the Teensy microcontroller, was used as a baseline as it presented the lowest measured latency values in previous literature. Tests labeled as "min" were situations where the ESP32 operated with the shortest possible path between a digital input trigger, and emission of the wireless message to the host. The "WiDig" configurations made use of the dual microcontroller setup, with the ESP32 performing the role of wireless and USB communication while the sensor acquisition was performed on the AtMega328.

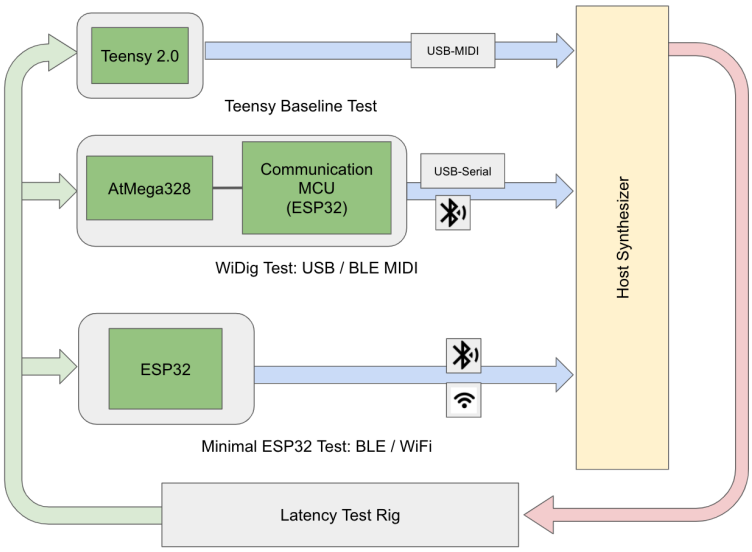


Fig. 4.12 WiDig end-to-end latency configurations.

Table 4.1 shows the measured end-to-end latencies for different test configurations, compared with some existing measurements from previous literature (McPherson, Jack, and Moro 2016). Based on the results, we see that it is possible to achieve the sub-10 ms la-

Table 4.1 End-to-end latency measurements. The WiDig and the product it replaces, the Wi-microDig, are presented in **bold**.

Connection	Latency	Std Dev
USB-MIDI (Teensy)	4.1 ms	0.4 ms
USB-MIDI (Teensy)*	5.1 ms	0.4 ms
USB-Serial (WiDig)	6.2 ms	0.35 ms
ESP32 BLE-MIDI (min)	7.5 ms	1.8 ms
ESP32 BLE-MIDI (WiDig)	**19.1 ms**	**2.7 ms**
ESP32 Wi-Fi RTP-MIDI	8.5 ms	8.0 ms
ESP32 Wi-Fi OSC (min)	7.6 ms	2.9 ms
Wi-Fi OSC*	6.7 ms	1.5 ms
Bluetooth 2.0-Serial (Wi-microDig)	**30 ms**	**14.6 ms**
BLE*	139 ms	21.9 ms

* indicates results from previous study (McPherson, Jack, and Moro 2016)

tency value for wireless interfaces using Wi-Fi and BLE, and is considerably faster than the previous I-CubeX wireless solution, the Wi-microDig operating via Bluetooth 2.0. The WiDig transmitting BLE-MIDI exhibited close to 20 ms of latency, which is improved from the 30 ms value of the Wi-microDig along with a significantly reduced jitter. The "minimal" BLE configuration was even faster and obtained an average latency of under 10 ms. This suggests that optimizations in the sensor acquisition and processing architecture can reduce the latency further. For example, the tested WiDig configuration involved passing the sensor data between two microcontrollers before transmission due to the existing sensor processing performed by the AtMega328. This eliminated the need to port the existing sensor processing code, but at the cost of running the processing on the slower AtMega328 compared to the ESP32. These additional processing steps could potentially be bypassed in a "low latency" mode of operation, or both the sensor processing and transmission could be implemented on the faster ESP32 in the future, potentially eliminating the two-microcontroller architecture altogether.

Another interesting finding is that the Wi-Fi implementations of OSC and Real-Time

Protocol (RTP)-MIDI were relatively close to each other. This is an expected result since both protocols make use of UDP as the transport layer, so whether protocol is used can be based solely on considerations for software compatibility.

Bandwidth Considerations

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 Wi-Fi links where possible. A test BLE application was implemented on the ESP32 and nRF51822[5] 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 2.2.1. Running a similar test on the ESP32 operating as an Wi-Fi OSC sender instead, the total bandwidth was considerably higher as seen in Chapter 4.1 that yielded around 220kbps for a single device.

Retrofitting MIDI Devices with BLE

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

[5]In the form of an RFDuino board running the same firmware via the supported Arduino libraries

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.

Fig. 4.13 The Yahamha MD-BT01 connected to a M-Audio MIDISport 2x2 USB-MIDI interface.

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 Port 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 4.14.

A simple MIDI timing application was written in C using the RtMidi library (Scavone

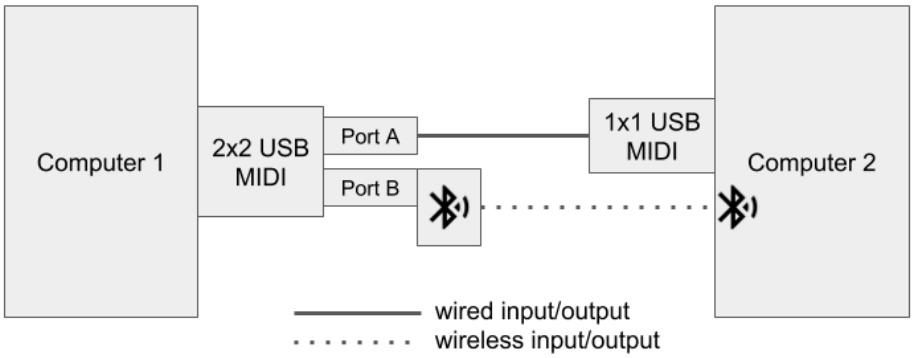


Fig. 4.14 Wired and BLE-MIDI test configuration.

and Cook 2004) in MacOS. 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 4.15 showing the operation of the sender and receiver modes.

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 500 ms, and the results are presented in Table 4.2, 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 26 ms of additional delay for the roundtrip, or about 13 ms for one way.

The roundtrip result, as shown in Table 4.2, shows a significant increase in latency when using the BLE-MIDI adapter compared to the wired interface of 29.9 ms and 4.3 ms,

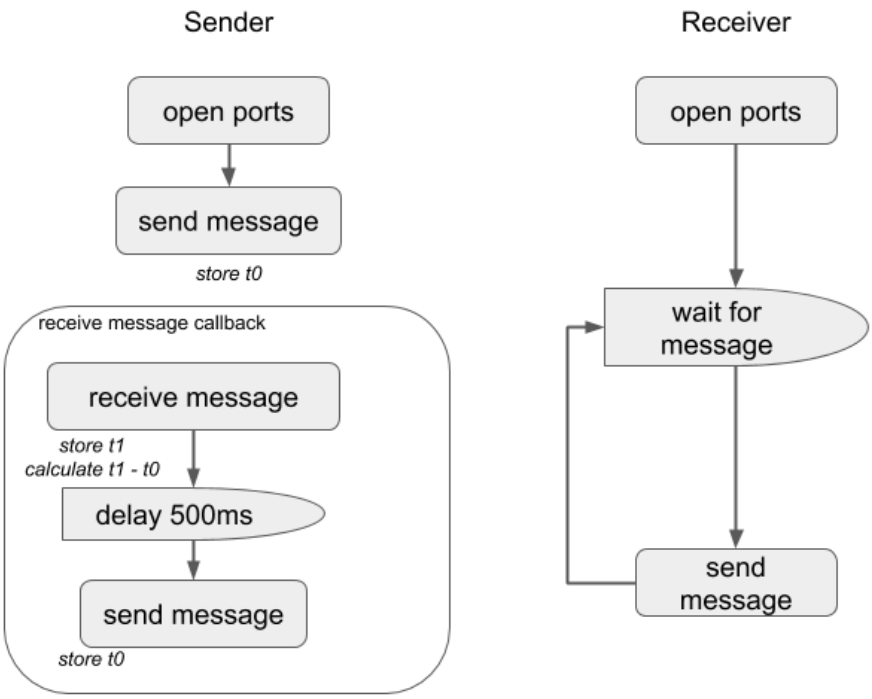


Fig. 4.15 Flow diagram of MIDI roundtrip latency measurement.

respectively. This suggests that these adapters may not be an ideal choice unless the wireless retrofitting is absolutely necessary and the increased latency is acceptable for a particular application.

Table 4.2 Round-trip latency 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

4.2.6 Summary

In this work we have described the testing of a number of sensor interface configurations with a focus on MIDI over BLE in the context of building a new wireless sensor interface for Infusion Systems. MIDI over BLE, MIDI and OSC over Wi-Fi, MIDI over USB, and a legacy Bluetooth 2.0 connection were tested for end-to-end latency. We also evaluated the performance of BLE-MIDI adapters. Overall, Wi-Fi exhibited the best latency performance for wireless interfaces with consistently below 10 ms average end-to-end latencies for both MIDI and OSC implementations, while BLE-MIDI performed slightly worse but was capable of sub-10 ms performance. Having established the performance characteristics of BLE-MIDI as capable of significantly reducing the latency compared to the previous Bluetooth 2.0 implementation of the Wi-microDig, we proceeded with the production and manufacturing of the new WiDig sensor interface. An added benefit of using the ESP32 platform for this design was the possibility to support Wi-Fi implementations as well.

4.3 Internet of Musical Things: Comparison of LoRa vs. BLE and Wi-Fi on the ESP32

4.3.1 Overview

The growth of mobile computational and sensing technologies along with the widespread availability of connection infrastructure of the internet has led to an increase to the number of sensor interfaces and devices that make up the Internet of Things (IoT) (Atzori, Iera, and Morabito 2010) on a global scale. Of particular interest, is how DMIs (Miranda and Wanderley 2006) contribute to this landscape through the use of such technologies for artistic expression and creative applications. By leveraging network connectivity DMIs can readily become a part of the Internet of Musical Things (IoMT) (Turchet et al. 2018) and easily span across the planet include the Global String (Tanaka and Bongers 2002), or concepts embodied by the Global Hyperorgan (Harlow 2018). At the same time, DMIs have specific requirements and are often not evaluated within its own context, and this work attempts to address the specific issue of protocol selection when implementing DMIs.

Rapidly growing and accessible open-source hardware and software ecosystems such as the Arduino (Kushner 2011) make it increasingly easier to implement sensor-based systems, and there is a growing number of such devices both in the research and commercial domains. Of particular interest are wireless interfaces, as they allow the implementation of portable and untethered controllers and open up the possibility of compact handheld devices and wearables. Advancements in wireless and battery technology (Cook 2009) make the replacement of wired solutions faster and more reliable, ameliorating some of the challenges faced previously (Cook 2001).

Like previous evaluations described thus far, the ESP32 was the base platform for the sensor interface in this investigation. A large number of development boards are available,

including many with additional LoRa hardware that provides support for an emerging standard that shows promise for IoT applications (Augustin et al. 2016). Figure 4.16 shows a ESP32 development board used for the evaluation described in this section.

The three protocols chosen for the evaluation are: MIDI messages over BLE, LoRa over a custom protocol (since no standards yet exist in this context), and OSC (M. Wright 2005) over Wi-Fi. The BLE- and Wi-Fi-based protocols were chosen due to their compatibility with commodity hardware, as most DMI designers use general-purpose computing systems and off-the-shelf operating systems. LoRa, as an emerging IoT protocol, is of interest in this context since it has yet to be applied in the context of DMIs, and looking at the underlying LoRa protocol itself provides an idea of the maximum performance before being subject to constraints of the stricter access and bandwidth limitations of LoRaWAN which builds on top of LoRa (Adelantado et al. 2017).

The three protocols chosen for evaluation in this work are not intended to be comprehensive, but serve to demonstrate the usage of the test system to reveal factors that may affect protocol selection in the context of a DMI. Additionally, the multi-protocol ESP32 development board allows the same underlying hardware to be used to eliminate differences between hardware implementation and isolates the protocols themselves for comparison.

The TTGO ESP32 LoRa development board (Figure 4.16) was used to implement the test sensor interface for this evaluation. This development board not only supports the native wireless capabilities of the ESP32 (BLE and Wi-Fi), but also adds a Semtech SX1276 IC that provides the LoRa radio. The ability to implement all three protocols on the same board provided a unique opportunity to test the three protocols using the same underlying microcontroller platform, and rules out potential differences due to underlying hardware.

In addition to the latency measurements similar to the investigations performed in

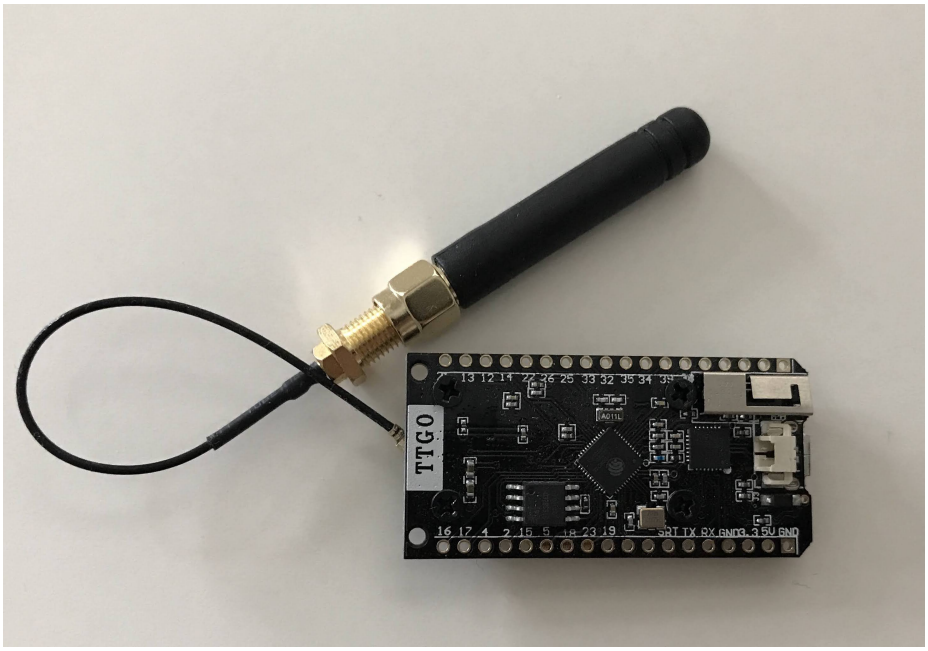


Fig. 4.16 The TTGO ESP32 LoRa development board.

Sections 4.1 and 4.2, we also added power measurements to observe the differences between each protocol and transmission configurations.

4.3.2 Evaluation Setup

Overall, the latency measurement configuration remains identical compared with the configuration described in Chapter 3. Figure 4.17 shows the system diagram of the test configuration. From previous tests described in literature as well as our own evaluations, there are no user configurable BLE or Wi-Fi settings to adjust for latency performance, but for LoRa a number of transmission parameters can be adjusted.

For LoRa transmissions there is a trade-off between airtime (which affects latency), range, and power (Adelantado et al. 2017). Increasing the Spreading Factor will increase the time on air and the power consumption as well as latency value, but improve the range and reliability of the transmission for a given bandwidth value. Conversely, increasing the

bandwidth will reduce airtime and power consumption, at the cost of range. Since in our evaluation we are not looking at range specifically but the minimal latency performance, we empirically found the value of 500 kHz bandwidth and Spreading Factor of 7 yielded the lowest latency by measuring different combinations of these values and observing the resultant latencies, and employed this configuration for the remainder of the tests.

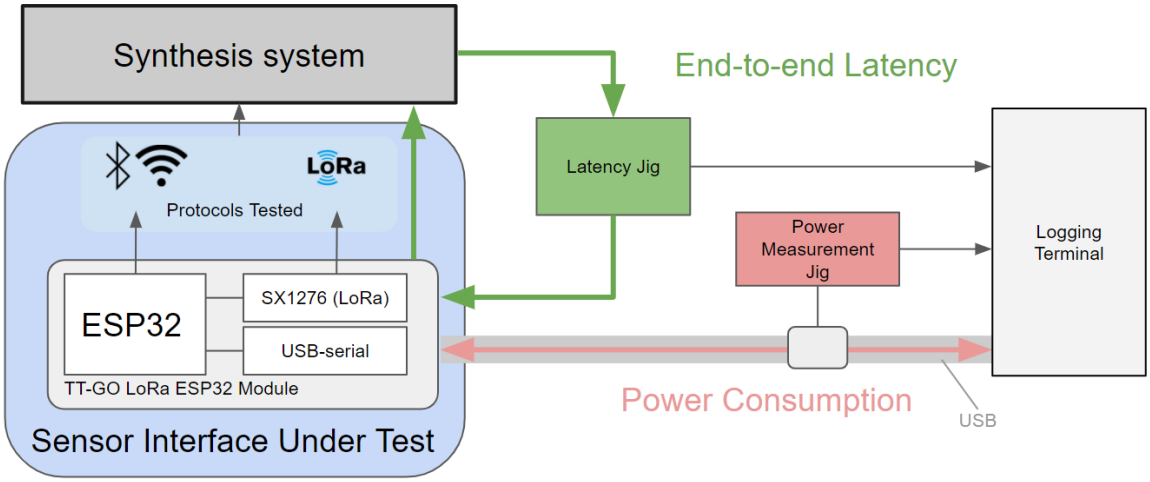


Fig. 4.17 Test system block diagram showing the power/latency measurement rigs and the sensor interface under test.

The Wi-Fi OSC and BLE MIDI implementations were identical to evaluations described in previous sections 4.1 and 4.2. However, for the LoRa configuration, because no host computer has native LoRa radio support, a second ESP32 LoRa module was used to implement a LoRa to USB-serial adapter operating at the standard rate of 115200 bits per second. While the serial connection will add a small amount of latency, in this situation the LoRa interface *must* go through this channel so it should be considered as part of the overall interface. We implemented a separate test consisting of just the USB-serial receiver module for reference, and triggered it directly similar to the wired serial tests described by (McPherson, Jack, and Moro 2016), as shown in Figure 4.18. The average latency was

measured to be 6.3 ms with the ESP32 module in this situation.

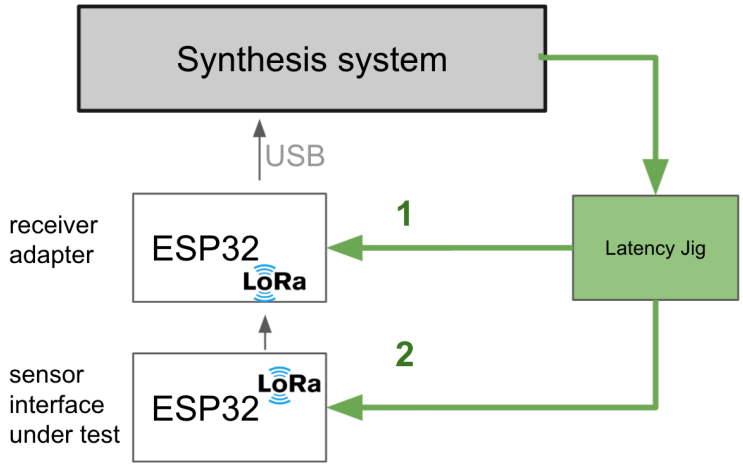


Fig. 4.18 Baseline test (1) of using a wired ESP32 triggered directly, compared with the wireless interface as tested for the remainder of the LoRa evaluations (2) where the wired ESP32 acts as the LoRa receiver for the host synthesis system

The main addition to the previously described evaluation configuration is a power measurement rig. Since the device under test is powered via a USB cable, we can measure power consumption using an in-line power meter. A power measurement rig, consisting of an ESP32 microcontroller and Texas Instruments i2c based INA219 power monitor IC was used to sample and transmit power consumption data (Texas Instruments 2011). A micro-USB cable was spliced so that the 5V USB line can be passed through the INA219 for high-side current monitoring (measuring the current between the positive power input terminal and the device). The built-in power consumption computation of the INA219 simplifies the logging firmware, which simply performs i2c bus reads and outputs the power values to the logging terminal via a USB serial interface. The power measurement jig was implemented using a Lolin D32 Pro ESP32 development board, for the simple reason that many were available in the lab for other projects. Figure 4.19 shows a block diagram of

the power measurement jig.

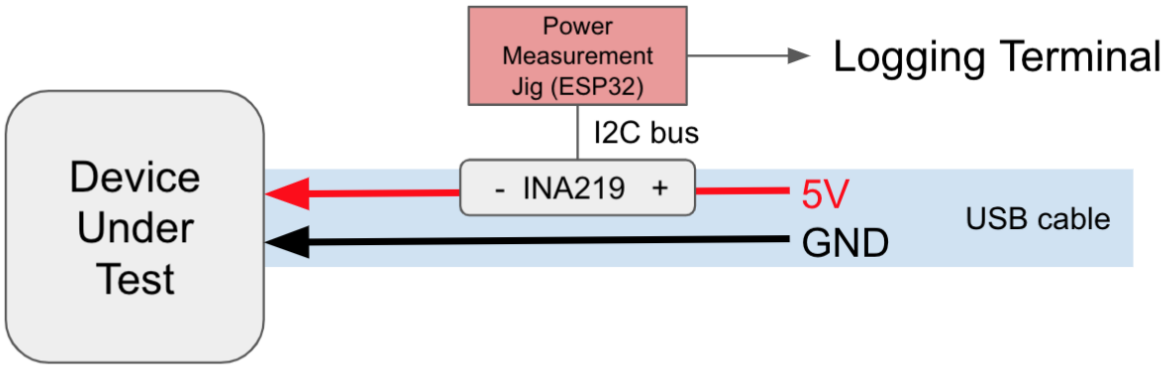


Fig. 4.19 Block diagram of the power measurement jig.

We ran two main sets of tests for the three protocols. In both cases, we were interested in the latency of single trigger events, but the sensor interfaces were set to operate either in *trigger* or *continuously sampling* modes, as described in 3.3. The intention of these two tests is to first see what the minimal latency is for the particular configuration, followed by a measurement of what the maximum sampling rate can be supported.

4.3.3 Trigger Test Results

Using the same latency testing procedure as previous examples, the power consumption and average latencies for each protocol, when the test triggers were sent at 4Hz, are presented in Figures 4.20 and 4.21. From this particular configuration it appears that OSC over Wi-Fi has the lowest latency, and the value of 5.9 ms is comparable to other results obtained in Sections 4.1 and 4.2. LoRa exhibited the second lowest latency values, and interestingly enough, the lowest power consumption as well.

The distribution of the latency values are presented in Figure 4.22 via an ECDF and indicate that the latency behaviour of BLE and LoRa are quite similar. In both cases, 90%

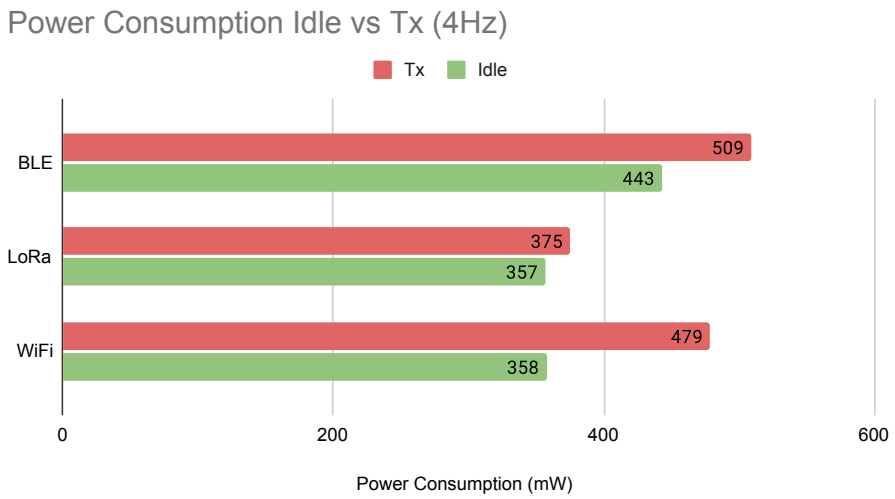


Fig. 4.20 The average power consumption when transmitting at 4 Hz vs. idle.

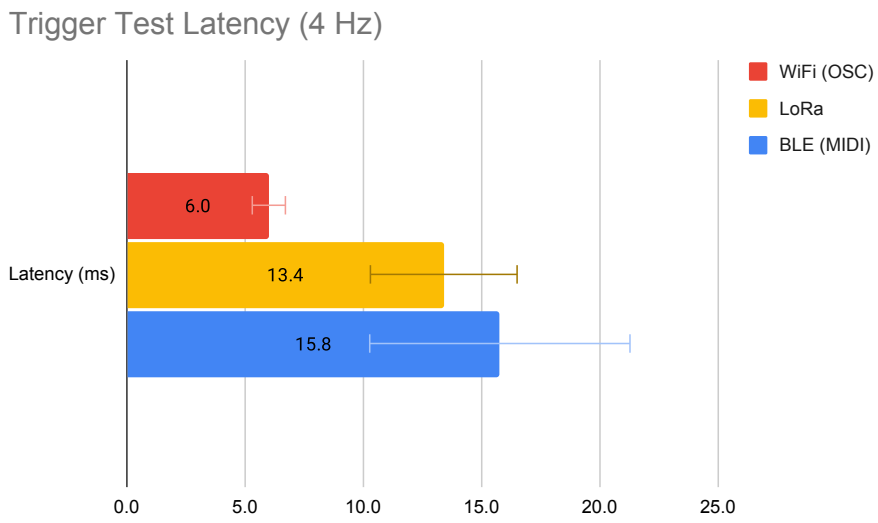


Fig. 4.21 The latency when transmitting at 4 Hz.

of latency values under 20 ms. Wi-Fi was the only protocol that managed 90% of latency values below the 10 ms threshold.

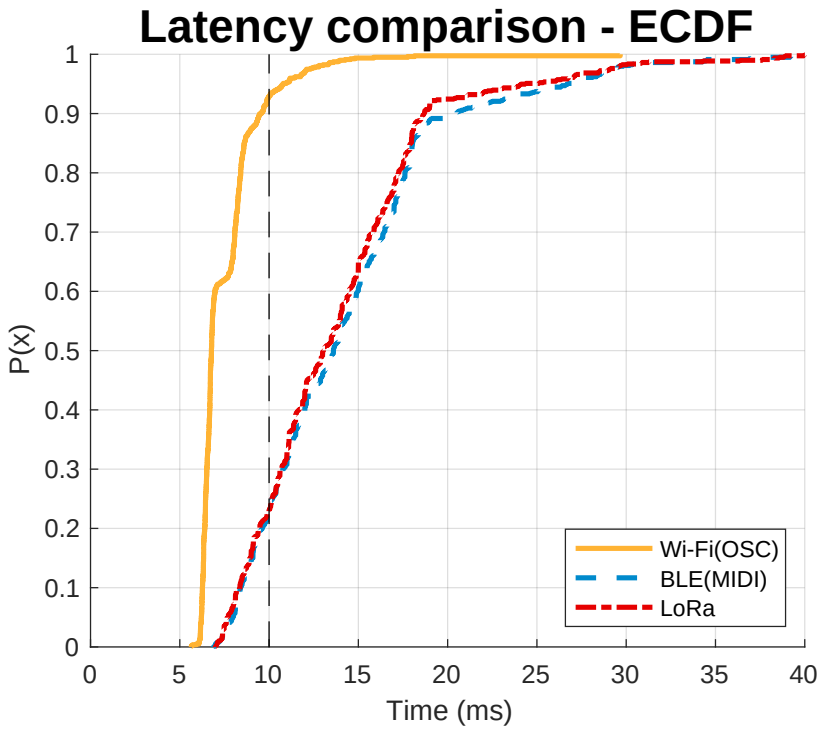


Fig. 4.22 ECDF of latency values, showing that Wi-Fi achieved over 90% of latency values below the 10 ms threshold compared with around 20% for BLE and LoRa.

4.3.4 Streaming Test Results

Similar to the transmission rate scaling for a single device, as explored in 4.1, in this evaluation we measure the latency behaviour of the protocols under test at different transmission rates. This test considers a continuously streaming output from the sensor interface, and allows us to identify the potential impact on latency performance as the transmission rate increases, and also to observe the maximum transmission rate possible with the particular hardware and protocol configuration.

The sensor interface firmware was configured to gradually increase the send rate every 30 seconds, and the latency jig issued test triggers at the same intervals (of 4 Hz). Table 4.3 shows the behaviour of the test firmware's send rate over time.

Table 4.3 Increasing the transmission rate over time.

Time (s)	Send Rate (Hz)	Loop time (ms)
0-30	10	100
31-60	20	50
61-90	25	40
91-120	50	20
121-150	75	13.3
151-180	80	12.5
181-210	100	10
211-240	200	5
241-270	400	2.5
271-300	inf*	minimal

* no additional delay in the loop means the system will attempt to transmit as quickly as possible

The results are presented in Figure 4.23 which shows the packets received per second[6], measured latency, and power consumption of the three protocols over the duration of the test.

From the measured results, both BLE and LoRa peaked at around 150 messages per second (at time $>$ 210 seconds in Figure 4.23a), while Wi-Fi was able to reach a ceiling of around 1400 packets per second (beyond the extents of the graph). While both BLE and LoRa achieved a maximum of around 150 messages per second, BLE experienced a severe increase in measured latency that suggested packets were being dropped. Considering that the minimal connection latency for BLE is 7.5 ms as presented in Chapter 2.2.1, this would correspond to a maximum rate of 133 Hz and suggests that packets were being dropped. LoRa, on the other hand, continued to transmit normally at this maximum speed and

[6]measured by a message counter in the synthesis patch that was polled at 1 second intervals

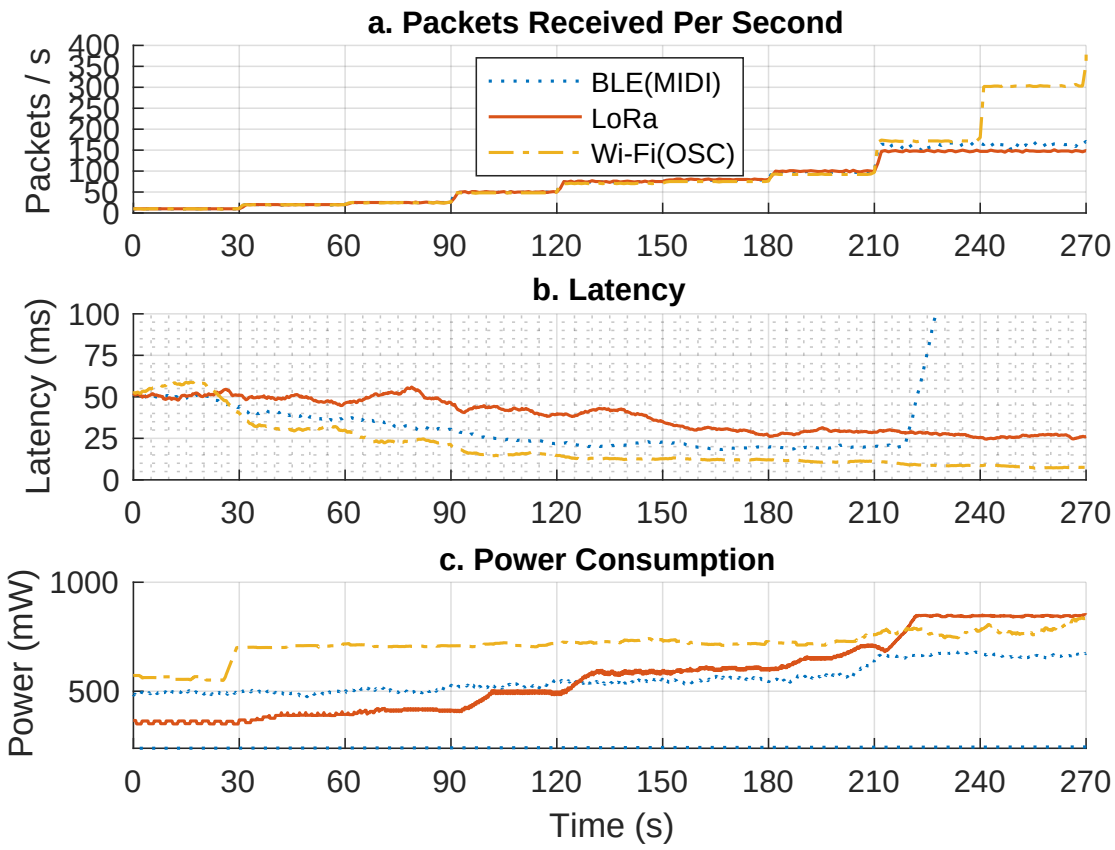


Fig. 4.23 Incrementally increasing transmission rate test to empirically determine the maximum continuous transmission rates.

the difference in behaviour is likely attributed to how the transmission rate limiting is performed in the radio firmware to throttle the transmission. While the LoRa to USB adapter would have contributed to the latency (as presented in Figure 4.18, the available bandwidth of the USB-Serial adapter (115.2 kbps) should not have impacted this rate.

Beyond 100 measured packets per second, BLE and LoRa did not see any improvements and the former became unusable with latencies exceeding 200 ms, likely due to network overload issues. While we generally observe latency values in Figure 4.23b that behaves according to the 1/2 sampling and transmission rate relationship as described in Section 3.3.3, the corresponding *maximum* transmission rates impose a *minimum* latency value for each protocol. The maximum transmission rate of Wi-Fi was found to be around 1400 Hz, and at that rate, the measured latency was close to the triggering test of around 7 ms which was similar to the trigger test results described in Section 4.3.3.

In terms of power consumption, the measured values are plotted in Figure 4.23c. At the maximum sending rate of 1400 Hz for Wi-Fi and around 150 Hz for BLE and LoRa, Wi-Fi exhibited the highest power consumption at 732 mW for all three sending rates while BLE and LoRa were less at 675 mW and 606 mW, respectively. At the highest usable rate for all three protocols of 100 Hz, the power consumption was 557 mW, 594 mW, and 727 mW for BLE, LoRa, and Wi-Fi, respectively. Curiously, it was possible to attempt to transmit LoRa messages at higher rates consuming more power, but without an effective increase in transmission rate or reduce the measured latency. However, since this was not considered as a usable configuration, this value is disregarded.

4.3.5 Summary

In this investigation the three protocols, MIDI over BLE, OSC over Wi-Fi and a custom, minimal protocol over LoRa were implemented and evaluated. While Wi-Fi appeared to

be the fastest measured solution and the only one to achieve end-to-end latency of below 10 ms, it also has the highest power consumption. For continuous streaming applications, BLE and LoRa exhibited an effective message rate of around 100 Hz and 150 Hz respectively when considering both the measured latency values as well as maximum number of effective packets received.

4.4 BLE Device and Transmission Rate Scaling

Having performed comparative tests of BLE performance in Sections 4.2 and 4.3, in this section we present an evaluation of BLE device and transmission rate scaling similar to the Wi-Fi tests of Section 4.1. Since the overall test configuration is similar to previous tests, we will mention specific differences and proceed directly to the results.

4.4.1 Overview

From the single device scaling results observed in Section 4.3, we note the breakdown of BLE performance somewhere between a transmission rate of 100 and 150 Hz. Using the theoretical 133 Hz that corresponds to the BLE connection interval of 7.5 ms, we acknowledge this as a potential point of interest and varied our evaluations to observe the effects on measured latency around this value.

4.4.2 Single Device Transmission Rate Scaling

Expecting that the theoretical breakdown point should be around 133 Hz, we first performed a single device transmission rate scaling test by varying the transmission rates across values around this point to see when a significant increase in large latency values are observed. Table 4.4 shows the inter-message times of the sensor interface firmware and the associated

transmission rate that ranging from 100 Hz to 140 Hz, and the measured average latency of 1000 triggers that were performed every 250 ms.

loop time	Send Rate	Average Latency	Std Dev.
10 000 µs	100 Hz	20.2 ms	10.7 ms
9000 µs	111 Hz	17.7 ms	7.1 ms
8333 µs	120 Hz	19.9 ms	8.2 ms
8000 µs	125 Hz	20.1 ms	12.2 ms
7800 µs	128 Hz	18.5 ms	6.8 ms
7500 µs	133 Hz	24.1 ms	17.3 ms
7000 µs	143 Hz	47.4 ms	53.6 ms
7500 µs	167 Hz	73.8 ms	54.1 ms

Table 4.4 Measured latency for increasing transmission rates of a single BLE device.

A graph of the average latency plotted against the transmission rate for a single device is presented in Figure 4.24, and shows a drastic increase around the 133 Hz (7.5 ms interval) transmission rate. This finding, along with the measurements presented in the previous section (4.3) is consistent with the theoretical value associated with the BLE connection interval.

4.4.3 Multiple Device Scaling

Having established the upper limits of the send rate for a single device, we measured the latency performance of one device while adding additional ones. The first test was performed at a per-device transmission rate of 125 Hz, and we observed a relatively similar measured latency performance as the single device test when adding up to 4 devices, as presented in Table 4.5. However, when the 5th device was added, we experienced a significant number of timeouts which suggested the configuration was inoperable.

We then reduced the individual transmission rate to see if it would be possible to support

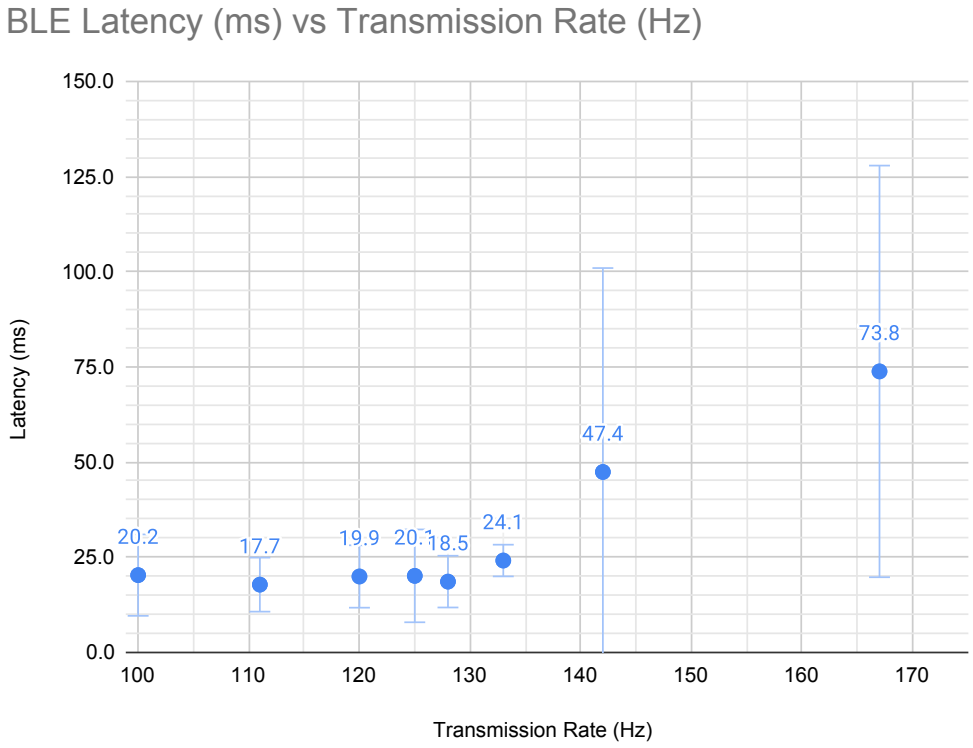


Fig. 4.24 Measured BLE Latency as a function of transmission rate for a single device.

# devices	Total Message Rate	Average Latency	Std Dev.
1	125 Hz	24.2 ms	17.0 ms
2	250 Hz	17.8 ms	6.7 ms
3	375 Hz	18.1 ms	7.5 ms
4	500 Hz	24.2 ms	17.0 ms

Table 4.5 Measured latency of device under test when adding devices operating at 125 Hz.

more devices, and using a transmission rate of 75 Hz, we observe the latencies presented in Table 4.6.

# devices	Total Message Rate	Average Latency	Std Dev.
1	75 Hz	17.5 ms	6.8 ms
2	150 Hz	19.7 ms	7.7 ms
3	225 Hz	20.4 ms	8.5 ms
4	300 Hz	23.5 ms	10.5 ms
5	375 Hz	23.8 ms	9.9 ms

Table 4.6 Measured latency of device under test when adding devices operating at 75 Hz.

While we did not reach a point where the measured latencies increased significantly, we were unable to add more than 5 devices. Finally, in order to eliminate the possibility of network congestion completely, we also attempted to reduce the transmission rate even further to just 10 Hz, and observed a similar outcome as shown in Table 4.7, where we were unable to pair more than 5 devices successfully.

4.4.4 Summary

In this section we presented the scaling characteristics of BLE-MIDI by measuring the latency of one device while altering the transmission rate as well as number of concurrent devices. Based on the measurements, we have identified the maximum transmission rate

# devices	Total Message Rate	Average Latency	Std Dev.
1	10 Hz	23.4 ms	11.9 ms
2	20 Hz	16.4 ms	6.1 ms
3	30 Hz	16.8 ms	6.7 ms
4	40 Hz	17.5 ms	9.1 ms
5	50 Hz	26.3 ms	14.6 ms

Table 4.7 Measured latency of device under test when adding devices operating at 10 Hz.

of a single device at close to the minimal transmission interval of BLE-MIDI of 7.5 ms or 133 Hz. Transmission rates below this value will lead to nominal performance, but when exceeded will lead to drastically increasing latencies and worse performance. However, it is possible to operate multiple devices close to this rate as evident by the device scaling tests, and here there were two constraints of either five concurrent devices or around 500 messages per second total. From the three sets of device scaling measurements performed at 125 Hz, 75 Hz, and 10 Hz per device, we are able to observe close to the optimal latency performance as long as we remain below both of these limits.

4.5 Collaborative Projects

4.5.1 Embedded Linux Platform Comparison

In this section we present the evaluation of embedded Linux platforms for real-time audio processing in the context of an augmented instrument, the GuitarAMI (Meneses, Freire, and Wanderley 2018). During this investigation, we used the same testing methodology to evaluate the end-to-end latencies of various system configurations to support the process of platform selection. The work was presented as a conference paper (Meneses et al. 2019) where the author of this thesis was second author. The contributions include the setting

up of a similar evaluation system as presented throughout this thesis, as well as helping with the collection and analysis of data.

Fig. 4.25 Three sets of the GuitarAMI consisting of the main module and remote sensor (Reproduced with permission).

The physical reduction in size of more capable processing platforms such as the Raspberry Pi and BeagleBone Black have resulted in open source platforms such as the Satellite Center for Computer Research in Music and Acoustics (CCRMA) (Berdahl and Ju 2011), Bela (McPherson 2017), or Prynth (Franco and Wanderley 2016a). These frameworks provide accessible hardware and software platforms that allow a complete instrument to be programmed and implemented within a single, compact and standalone embedded device and eliminates the need for an external processing computer.

The GuitarAMI, as shown in Figure 4.25, is an example of a DMI that makes use of such an architecture. In the case of the GuitarAMI, the entire sensor mapping, audio processing and synthesis system fits within a small "stomp-box" as shown by the larger boxes in Figure 4.25. The embedded Linux processor is contained within the main module, while the smaller wireless remote unit contains an IMU and ultrasonic distance sensor that

is mounted on the neck of the guitar to track the motion of the instrument as well as hand movements.

As an Augmented Musical Instrument (AMI) or *hyperinstrument* (Machover 1992), the GuitarAMI processes both signals from gestural sensors placed on the instrument or worn by the performer, as well as audio input from the acoustic instrument itself. In the case of the GuitarAMI, the nylon string acoustic sound is picked up using contact microphones mounted on the body of the instrument and connected to the Analog-to-Digital Converter (ADC) input of the sound card on the embedded processing module. In such a *hyperinstrument*, the resultant audio produced by the system could consist of purely synthesized sounds, or a modified version of the acoustic sound, or some combination of both.

One of the design decisions was to choose a specific open source embedded Linux platform, and the latency performance was one metric of interest. Additionally, it was of interest to compare the performance of both wired and wireless interfaces to identify any potential differences between these two options.

Evaluation Setup and Results

In the evaluation, three candidate embedded Linux platforms, Bela (McPherson 2017), Prynth (Franco and Wanderley 2016a), and a custom processing unit implemented from a scratch using a stock Raspberry Pi (Raspbian) distribution, henceforth referred to as the "SPU" were tested. For each of these platforms, three types of signal paths were tested including audio-audio, signal-audio (via wired or built in sensor interface on the embedded hardware), and signal-audio via a wireless microcontroller communicating using OSC over Wi-Fi, as shown in Figure 4.26. Note that in the case of the audio only test, the test trigger emitted by the latency jig is sent directly into the sound card input which then directly

passes through the audio input, in a similar manner as previous literature that measured audio processing latency (MacMillan, Droettboom, and Fujinaga 2001).

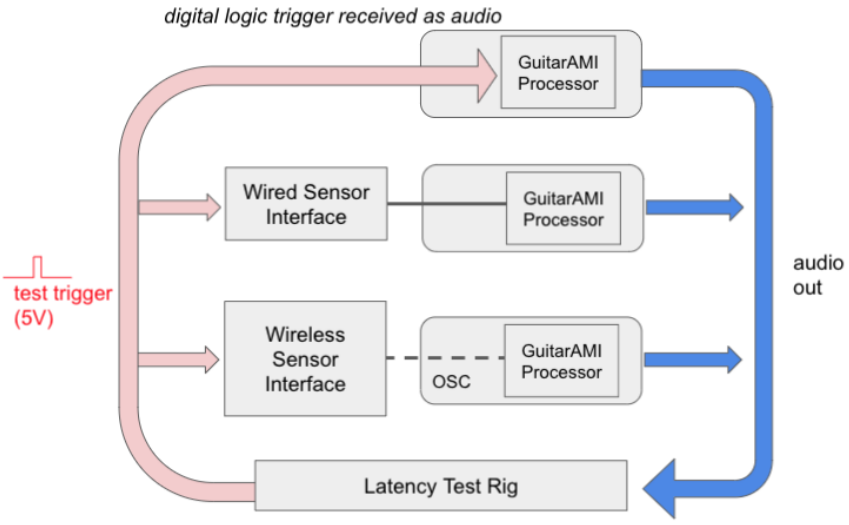


Fig. 4.26 Block diagram of test system.

The same latency measurement jig (Chapter 3) was used to measure both sensor input latency (for wired and wireless sensors), as well as latency of the audio processing system. The audio synthesis environment used was SuperCollider since the project had existing software implementations within that environment, and the audio configuration was set to 128 block size at 44.1kHz. For the sensor to audio output test, the framework was set to emit a reverse ramp signal via the audio output upon the recipient of the input trigger, similar to the Max/MSP patch as described in Chapter 3.2.

For each platform, Bela, Prynth, and the SPU, there was a number of potential wired interfacing options and they are presented in Table 4.8, along with the best performing interface for that platform (Meneses et al. 2019).

The above configurations show that the Bela platform functions as intended, with very fast direct access to the built-in input pins of the Beaglebone board that was used to

Platform	Wired Interface	Fastest
Bela	Analog Input pins*	x
Bela	USB-MIDI Input	
Bela	Onboard Universal Asynchronous Receiver-Transmitter (UART)	
Bela	USB UART	
Prynth	USB-MIDI Input	
Prynth	Onboard UART	x
Prynth	USB UART	
SPU	USB-MIDI Input	
SPU	Onboard UART	x
SPU	USB UART	

Table 4.8 Wired sensor interfaces available on each platform. * Denotes the use of the built-in analog input pins, while the remainder of the configurations employed an external microcontroller connected to the associated ports to facilitate communication.

implement the system (McPherson 2017). On the other Raspberry Pi based platforms, the fastest wired input configurations make use of the on-board UART that bypasses the USB bus, which can be the source of additional delays.

In Figure 4.27 the audio-in to audio out, and wired/wireless input to audio out latencies are presented.

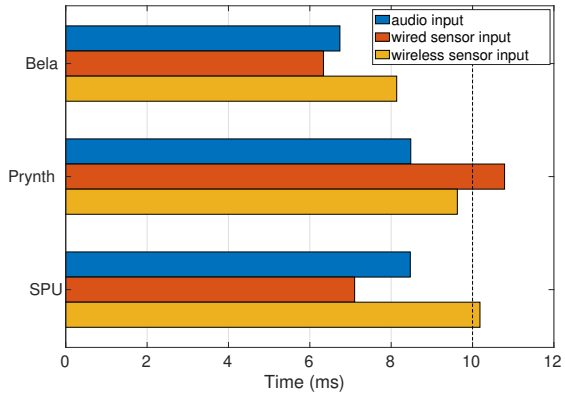


Fig. 4.27 Average latency measurements for each test configuration. For the wired sensor input values, the "Fastest" configurations as presented in Table 4.8 were used.

In the ECDFs, presented in Figure 4.28 and 4.29, we note the step-wise behaviour of the distribution, which are around 3 ms. This interval corresponds well to the 128 sample buffer at 44.1kHz, which yields a period of 2.9 ms per buffer. The consequence of this is that while the overall system latency behaviour is likely a smooth distribution when plotted against the measurement resolution, the discrete periods pertaining to the duration of the audio buffer is revealed when the test triggers are initiated at a constant interval between each measurement. While it appears that the majority of latency values (around 90%) were within the 10 ms threshold for nearly all of the configurations, almost all configurations exhibited latencies above 10 ms, especially in the case of wireless interfaces. The other noticeable behaviour is the wired interface latency of Prynth which shows 10% of the values greatly exceeding the 10 ms threshold, which may be due to the serial port implementation of that particular platform and warrants further investigation.

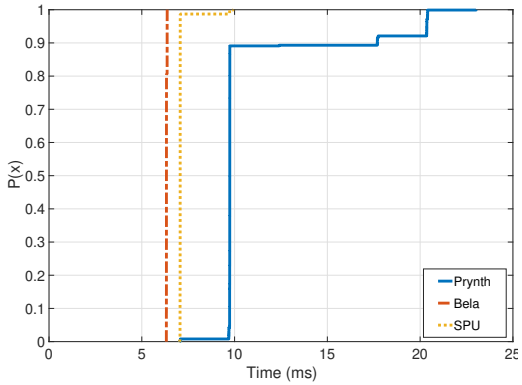


Fig. 4.28 ECDF of wired interface test.

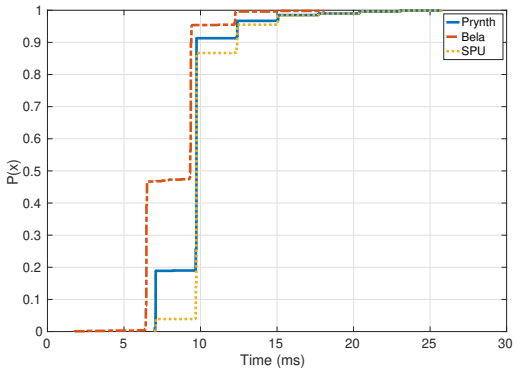


Fig. 4.29 ECDF of wireless test.

Beyond the latency values, this work also investigated the CPU load of different configurations and discussed some of the other implementation details such as the ease of software configuration, and other system limitations (such as built-in vs. add-on hardware required, and software limitations in the case of the Bela platform) (Meneses et al. 2019).

Summary

In this work, we applied our end-to-end latency measurement system to aid in the platform selection process for the continued development of an AMI, the GuitarAMI, and we were able to quickly and consistently evaluate a wide variety of sensor interface and audio configurations to characterize the performance of three embedded Linux-based systems for implementing DMIs. Overall, while the evaluation made use of the ESP32 microcontroller for the wireless sensor interface test for all platforms, the actual audio synthesis system employed is quite different from the configuration in previous chapters so the measured results are not directly comparable. However, the overall evaluation procedure and the observed trends are consistent with previous results, and demonstrated the adaptability and reuse of the measurement system.

4.5.2 Hybrid Latency Testing for Mocap Systems

In this section, we present the latency measurement of a hybrid Motion Capture (mocap) system consisting of an optical camera based system and a wireless IMU interface. The work was presented in a publication (Santos et al. 2021) where the author of this thesis (2nd author) helped with the adaptation and design of the latency evaluation system and subsequent analysis of the data.

The context of this evaluation was to compare the feasibility of using an inexpensive and simpler wireless IMU sensor setup versus a full fledged optical multi-camera mocap system[7] for the purpose of gait analysis by the first author of the aforementioned publication. The motivation to use such a system not only reduces the cost, but also reduces the complexity and requirements of the experiment deployment procedure that eliminates the mounting

[7] www.qualisys.com/

of cameras and equipment, attachment of Infra-Red (IR)-reflective markers on the subject, and dedicated processing hardware. A test configuration was constructed to capture data from both the mocap processor and the wireless sensor to get an initial idea of the latency introduced by each component.

Evaluation Setup and Results

To measure the latency of the mocap and wireless sensor, a "ground truth" signal was implemented by the means of a mechanical clap-board that contained optical markers as well as an electrical switch. This allowed to synchronization of two signals associated with the physical event: the captured minimal position of the IR markers as detected by the optical mocap system, along with the electrical contact between conductive surfaces of the clap-board, functioning as a switch. The latter signal is then fed into the latency measurement jig as the starting time and input to the wireless sensor interface, and the subsequently processed result from the mocap system as well as the wireless sensor can then be compared. Figure 4.30 shows a block diagram of the overall test system.

The general signal flow for either the mocap or wireless sensor interface configuration involves the transmission of a signal corresponding to the physical event, either via the position of the IR markers, or through the triggering of the wireless microcontroller, respectively. The mocap system is connected via ethernet to the processing computer, while the wireless sensor interface is connected via Wi-Fi and transmits OSC packets. Figure 4.31 shows the measured latency and distribution of the Wi-Fi sensor, while Figure 4.32 present the results of the mocap system.

One of the key findings of this evaluation was that in the tested configuration, the wireless sensor interface and Qualisys mocap system exhibited a comparable amount of latency, at around 23 ms. However, there was a significant amount of variation in the Wi-

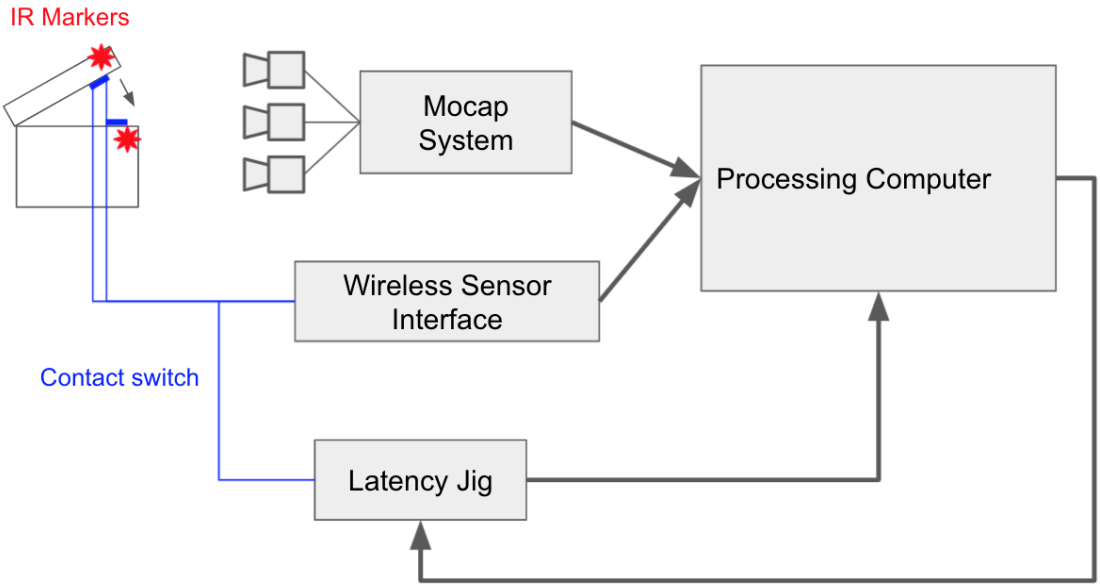
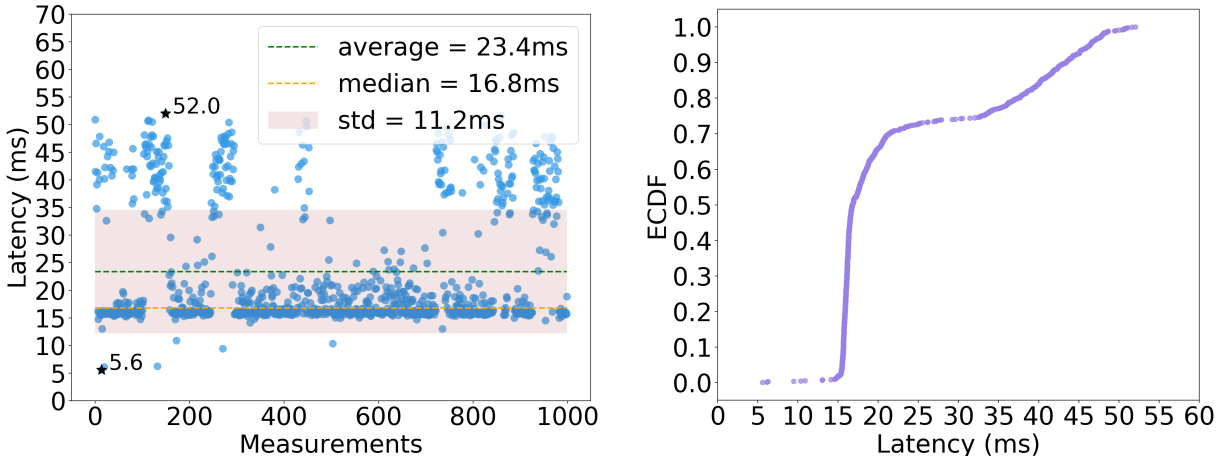


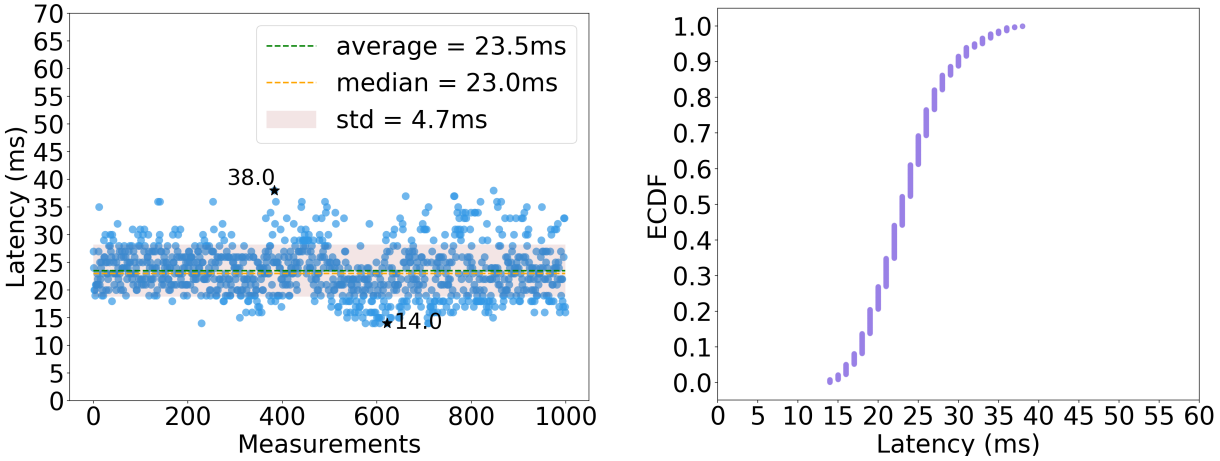
Fig. 4.30 Block diagram of the mocap/wireless latency measurement system.



(a) Latency distribution of Wi-Fi communication.

(b) ECDF of Wi-Fi communication.

Fig. 4.31 Wi-Fi Sensor Interface Measurements.



(a) Latency distribution of the latency testing of the mocap system.

(b) ECDF of the latency testing of mocap system.

Fig. 4.32 Mocap system latency measurements.

Fi measurements, and part of the issue is the networking environment where the experiment took place (weekday, in an office environment)[8]. The latency distribution of the wireless measurements appear to be more spread out compared to some of the other measurements presented in previous sections.

Summary

In this collaborative project, we demonstrated the adaptation of the latency measurement rig and general evaluation procedure applied to a hybrid system comprised of both a wired mocap system as well as a wireless sensor interface to compare the processing latencies of the two systems. We found the latencies between the two systems to be comparable, although the wireless interface exhibited significantly larger variance compared to previous measurements which suggests the effect of network activity can be significant on the latency performance of wireless interfaces.

[8]Even though there were no other wireless devices on the local router, in such an environment there are other access points operating on the same channels.

4.6 Test Trigger Interval and Impact on Measurements

Through our usage of the test configuration as described in Section 3.1, we identified an issue regarding the fixed test trigger delay intervals that introduced a potential phase dependency between any discrete processing intervals described above and the measured latency. Specifically, this issue has some impact on the accuracy of the measurements for some of the tests, and also the design of streaming-based interfaces that make use of protocols such as BLE with large connection intervals.

To illustrate this, we performed two trial measurements for BLE-MIDI where the connection interval was known to be 11.25 ms as configured in the BLE configuration of the ESP32 firmware, and confirmed with Apple's Bluetooth PacketLogger utility when the device was connected to the host. During the test, we repeated 100 test triggers for each *trigger interval* that increased from 95 ms to 130 ms at 1 ms intervals. In the first trial the *trigger intervals* were fixed, so the total time between trials was exactly the fixed delay plus the total system latency. In the second trail we introduced a +/- 25 ms random variation around the test trigger interval, which leads to the same *average* delay for the 100 triggers. As an example, when the *trigger interval* was 95 ms, every single test used the exactly 95 ms for all 100 tests for the fixed case while in the randomized case, they varied from 70 to 120, but were 95 on average. For both test cases we see that the two overall averages across all the points is around 14 ms, but the fixed version or synchronized version has a significantly larger variation between the largest and lowest measured latency values that is periodic in nature and is dependent on the test trigger interval used. The period of the variation also corresponds directly to the BLE connection interval of 11.25 ms as confirmed by Apple's "PacketLogger" utility. This introduces a flaw in the test as how often the test is run should not have an impact over the measurement result. This effect is presented in

Figure 4.33 where each plotted data point shows the average latency over 100 test triggers for a given *trigger interval*.

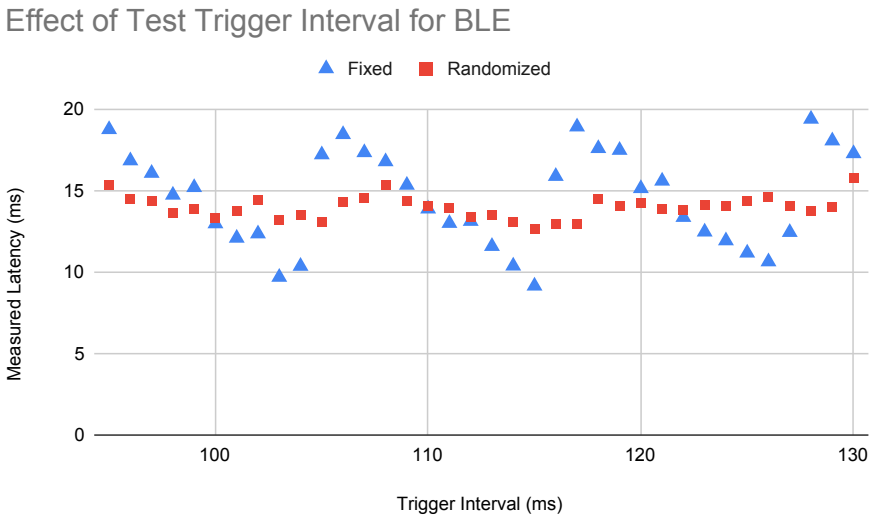


Fig. 4.33 Effect of randomizing vs. fixed test trigger intervals. Each point is the average of 100 samples at one particular test trigger interval.

This effect was not seen for Wi-Fi or LoRa during the measurements conducted in our various tests, regardless of whether the trigger interval was randomized or not around the average test trigger interval and show no periodicity as the test interval changed. Wi-Fi has much smaller connection/transmission intervals that are likely below the threshold of measurement, while LoRa does not impose transmission start times that synchronize the transmission to fixed intervals, but rather start the transmission as soon as possible. This observation implies that any previous measurements of protocols using this methodology with large synchronized connection intervals such as BLE, as presented by (McPherson, Jack, and Moro 2016) are slightly less accurate due to this phenomena not accounted for. The exact error in this situation could be as high as half the connection interval of the protocol. For example, in Figure 4.33 we can see that the average latency across all

measurements should be slightly under 15 ms, but without any variation in the synchronized case, one could be measuring average latencies as low as under 10 ms or as high as 20 ms, depending on what fixed delay interval was chosen. Similarly, for the results we presented in (J. Wang, Mulder, and Wanderley 2019), the actual minimal BLE measurements may in fact be slightly different from the true average value had the test triggers not been synchronized with the transmission windows.

Overall, this situation is simply a limitation in the specific implementation of the test jig, and the addition of a slight random value to the fixed delay interval between test triggers can eliminate this situation, and also create a more realistic test situation as real-life signals should not have any synchronization with the measurement device. Another potential way of decoupling this synchronization is for the test jig to be either triggered externally via an independent timer, or to use a time elapsed counter between each successive triggers to provide a true fixed interval.

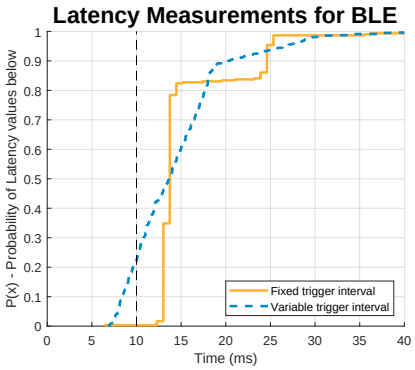


Fig. 4.34 ECDF of BLE for fixed (synchronized) and variable (unsynchronized) test trigger intervals.

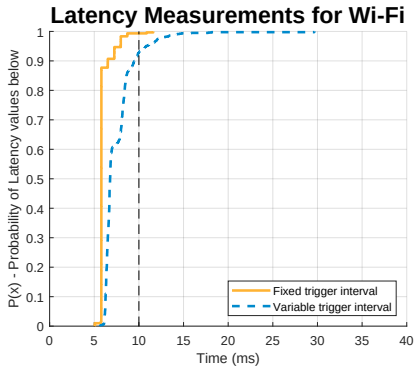


Fig. 4.35 ECDF of Wi-Fi for fixed (synchronized) and variable (unsynchronized) test trigger intervals.

On the other hand, if the transmission protocol does not have the long connection intervals similar to BLE, this effect does not exist even with this synchronization present in the test jig. Additionally, one interesting feature when operating the test jig in this

synchronized manner, is that it reveals any potential discrete blocks in the system (provided that it is above the measurement resolution). Figure 4.34 shows the ECDF of two sets of measurements of BLE-MIDI for a given test trigger interval, which can be thought of as pair of data points presented in Figure 4.33. The synchronized (fixed) version reveals "steps" in the distribution, where the smaller steps reveal the intervals corresponding to the audio buffer vector size of 32 (around 0.7 ms, similar to the blocks seen in the Wi-Fi scaling or Embedded Linux platform measurements of Chapter 4.1 and Chapter 4.5.1, respectively). We also see the much larger step of around 11 ms corresponding to the BLE connection interval. For Wi-Fi, we only observe the smaller steps of the audio buffer.

4.7 Summary of Findings

4.7.1 Minimal Latency

From our evaluations, Wi-Fi in general presented the lowest end-to-end latency for a wireless sensor interface from the perspective of a physical interface, and the average value for the best-case situations was around 7 ms, and the results were comparable between MIDI and OSC when transmitted through Wi-Fi as presented in Chapter 4.2. For BLE the measured latencies varied from as low as 7.5 ms in Chapter 4.2, to 15.8 ms in Chapter 4.3.

4.7.2 Maximum Transmission Rate

From the evaluations presented in Chapters 4.1, 4.3, and 4.4 we observed a maximum transmission rate of a little over 2300 Hz for Wi-Fi while still observing consistent and low system latency values. For BLE the number of messages that can be transmitted is limited by the connection interval, and while the lowest possible connection interval of 7.5 ms yields a theoretical transmission rate of 133.3 Hz (MMA 2015), which was empirically found to

be the upper limit in the evaluation presented in Chapter 4.4. Of course, for non real-time streaming applications (such as the bulk transmission of data for later/off-line analysis), it would be possible to bundle multiple messages up to 20 bytes in total length (MMA 2015) within a connection interval so that the effective rate received is higher. Similar to BLE, a protocol like LoRa, with it lower bandwidth and subsequent longer airtime, we observe similar transmission rate issues although the measured packets per second was slightly higher, at 150 Hz.

4.7.3 Maximum Concurrent Devices

Regardless of the interface, a single wireless channel saturates relatively quickly once additional devices are using the network. A seen by the Wi-Fi scaling test presented in 4.1 even with a few devices the latency increases significantly. For the 10 ms total latency threshold, with a single device transmitting at 100 Hz 90% of values are below the threshold, and this value drops about 10% for each two additional devices. This result suggests that multiple concurrent wireless channels may be necessary if both low latency and high sampling rates are required. In the case of BLE, where devices only have limited and relatively large connection intervals to transmit, this limit is lower and also bound by the physical number of devices that can be connected, which was observed to be five devices maximum regardless of total message transmission rate. However, this limit may be specific to the operating system and BLE implementation, and further investigation is needed to understand the cause of this limit.

Chapter 5

Usability of Sensor Interfaces and Mapping Tools

In this chapter we describe the usability of sensor interfaces by looking at physical hardware and driver compatibility, as well as protocol implementations that define how devices initiate connections. Then, we present work on mapping tools and frameworks and how the findings of this thesis presented thus far can be incorporated into such tools to provide useful features to support DMI design. Finally, we present how these findings can be applied to the T-Stick instrument that has seen on-going development at the Input Devices and Music Interaction Laboratory (IDMIL) for over the past fifteen years.

5.1 Compatibility and Usability

In this section we present an overview of the process by which the data from a sensor interface is delivered to the end user application.

5.1.1 Platform Compatibility

From a hardware perspective, most modern mobile devices have Wi-Fi and BLE capabilities, and desktop computers without built-in support for the wireless interfaces can be upgraded with add-on cards or USB-based dongles, including LoRa interfaces, as we have done in the system described in Chapter 4.3. However, for mobile devices, it is not as straightforward to add LoRa or additional RF hardware receivers directly. If the devices are meant to work with standard commodity computing hardware (desktops, laptops, smartphones, and tablets) without any additional physical add-ons, BLE and Wi-Fi implementations are more suitable. If there are no constraints against the addition of hardware adapters or bridges, then any potential wireless protocol could be added to a system. One other potential solution is to make use of adapters that implement a non-native wireless interface, and then bridge it to an existing physical interface such as USB, or the BLE-MIDI adapters described in Chapter 4.2.5. Of course, the potential added latency of such a setup would need to be verified.

In addition to the actual hardware, the associated device driver must be present as well. On desktop platforms it is not usually an issue to install custom driver software, but the process may not be as feasible for mobile platforms. Table 5.1 presents a list of interface and protocol combinations evaluated in this thesis, along with the hardware and software compatibility on current operating systems.

5.1.2 Connection Procedure and Software Integration

How devices start up and join a network (or re-join after a power loss) can also impact the usability of an interface. As an example, BLE-MIDI device must first be "paired" with the operating system, after which a virtual MIDI-compatible port will appear. Henceforth,

Physical Interface	*Messaging Protocol*	**iOS**	**Android**	**macOS**	**Windows**	**Linux**
Wi-Fi	OSC	None	None	None	None	None
BLE	MIDI	None	None	None	None*	None
Bluetooth 2.0	MIDI	N/A	S	S	S	S
LoRa	custom	H,S	H,S	H,S	H,S	H,S

Table 5.1 Operating system constraints of wireless sensor interfaces. S/H: additional (S)oftware drivers/(H)ardware interfaces required. None: native/built-in support. * While Windows supports BLE-MIDI ports by default, the ports themselves are not transparently exposed to software via the same MIDI API so the concept of a "BLE-MIDI" port is not the same as a traditional MIDI port, which means legacy software will not be able to use it.

any software that supports MIDI will be able to transparently make use of the device via a standard MIDI port. The only exception is in the Windows Operating System which, as of mid-2021, supports BLE-MIDI devices but exposes them via a separate API to user applications compared with "classic" MIDI devices. The consequence is that a bridging application is needed to interoperate with MIDI software (J. Wang, Mulder, and Wanderley 2019). For OSC over Wi-Fi, the device must be associated with a Wi-Fi network (via configuration of login credentials), and after which, the destination IP and port specified. In this situation, one potential issues is that the Dynamic Host Configuration Protocol (DHCP) server operating on the router should always assign the same IP address to the receiver host, to insure that the sender is able to transmit to the destination.

For LoRa, predefined LoRaWAN endpoints and gateways (Augustin et al. 2016) is a common approach to managing devices that may power on and off as needed, depending on power consumption requirements.

The implication of the connection/pairing procedure affects how devices can join/leave the network. For example, with an explicit pairing process of BLE, additional tools may be needed for management and automation, including the re-pairing and re-connection to the MIDI port. On the other hand, a Wi-Fi or LoRa device can simply turn on, automatically

join the associated network from previously stored settings and start transmission, since the receiving end is operating as a server waiting for connections at all times.

Once the data from the sensor interface is available in the host system, it can be consumed by the end-user application. Here, how the application makes use of the data is dependent on the actual application. In our evaluation we used Max/MSP to receive and process the data to synthesize a single trigger output (as described in Chapter 3), but in real life the actual mapping may be more sophisticated. The subsequent section presents the process of mapping and mapping tools in more detail.

5.2 Mapping Tools and Frameworks

For a DMI, where it is possible to make arbitrary associations between the input signals and resultant output, mapping is a strong determinant of the behaviour of the instrument (Rovan et al. 1997). Research on mapping has grown steadily over the last two decades, with the proposition of tools and environments to facilitate the design of mapping strategies for DMIs (Wanderley 2002), (Wanderley and Malloch 2014). At IDMIL we have been developing mapping tools and frameworks for over a decade through collaborative interdisciplinary projects, and in this section we present how some of the findings in the interface evaluations described thus far in this thesis may apply in this context. In this section, we first provide a brief overview of the libmapper framework that was developed to support the design of mappings. Then, we describe some of the tools that were built to make use of libmapper. Finally, we present some potential ways that the results of our evaluation of sensor interfaces could be incorporated into the existing tools to further support the mapping design process.

Visual Mapping Tools and Environments

There are a number of tools and frameworks that support DMI design. Some, such as MaxMSP[1], Pd[2], are full programming environments that can be used to define the structure of the entire instrument. There are also various toolboxes dedicated to mapping specifically, and they include standalone applications such as OSCulator[3] and junXion[4], or Wekinator (Fiebrink, Trueman, and Cook 2009), which provide mapping environments that specifically receive input signals, apply a defined set of mapping operations, and finally output the result to preset destinations. Other toolboxes are designed to work within a particular environment such as Max/MSP, and they provide objects representing mapping and signal conditioning primitives (Steiner 2006), or matrix-based manipulations specifically for mapping (Bevilacqua, Müller, and Schnell 2005).

Combinations of these tools allow visual representation of the connection and processing of signals that make up part of the mapping process, as well as implementation of interfaces to hardware and software components including input devices and synthesis systems. However, none of these offer the manipulation of mapping between devices and signals as an individual entity. Additionally, through collaborative contexts such as working with musicians and composers in various settings (Ferguson and Wanderley 2010), a number of features absent from existing tools were identified (Malloch, Sinclair, and Wanderley 2014), including:

- Providing free and re-configurable mappings to aid in experimentation and exploration

[1] https://cycling74.com/products/max/
[2] https://puredata.info/
[3] https://osculator.net/
[4] http://steim.org/product/junxion/

- Providing compatibility between devices and standards to avoid the need for manual and explicit connection implementation
- Providing the ability to freely use interesting mapping layers between devices, including some of the tools described above

In the following section, we present libmapper, a mapping framework and supporting tools that implement these features. In general, libmapper and its associated applications can be considered more generally within the realm of creativity support tools, and many of the goals including the facilitation of exploration, experimentation, and collaboration in open-ended activities that lead to creation of novel artifacts and outcomes (Shneiderman et al. 2006).

libmapper

libmapper (Malloch, Sinclair, and Wanderley 2014) is a distributed software library to support the creation of mappings between digital signals in an interactive context. Conceived through needs arisen during a number of collaborative interdisciplinary projects around DMIs design and implementation, libmapper was designed to support a network of connected devices that capture and consume control data, and in generally can be used for real-time interactive multimedia applications beyond musical instruments. The main features of libmapper include:

- *Automated discovery of devices and signals.* Instead of explicitly identifying and connecting devices and endpoints, a libmapper device will automatically advertise itself on the network along with the signals it contains. This avoids the need to keep track of IP addresses and ports as described for explicit OSC endpoints in Section 5.1.2.

- *Distributed and concurrent creation of connections and mappings.* Connections and maps between signals can be created and observed on the fly, from multiple users on the network. This allows rapid changes to be made without having to re-code the connections and mapping behaviour.

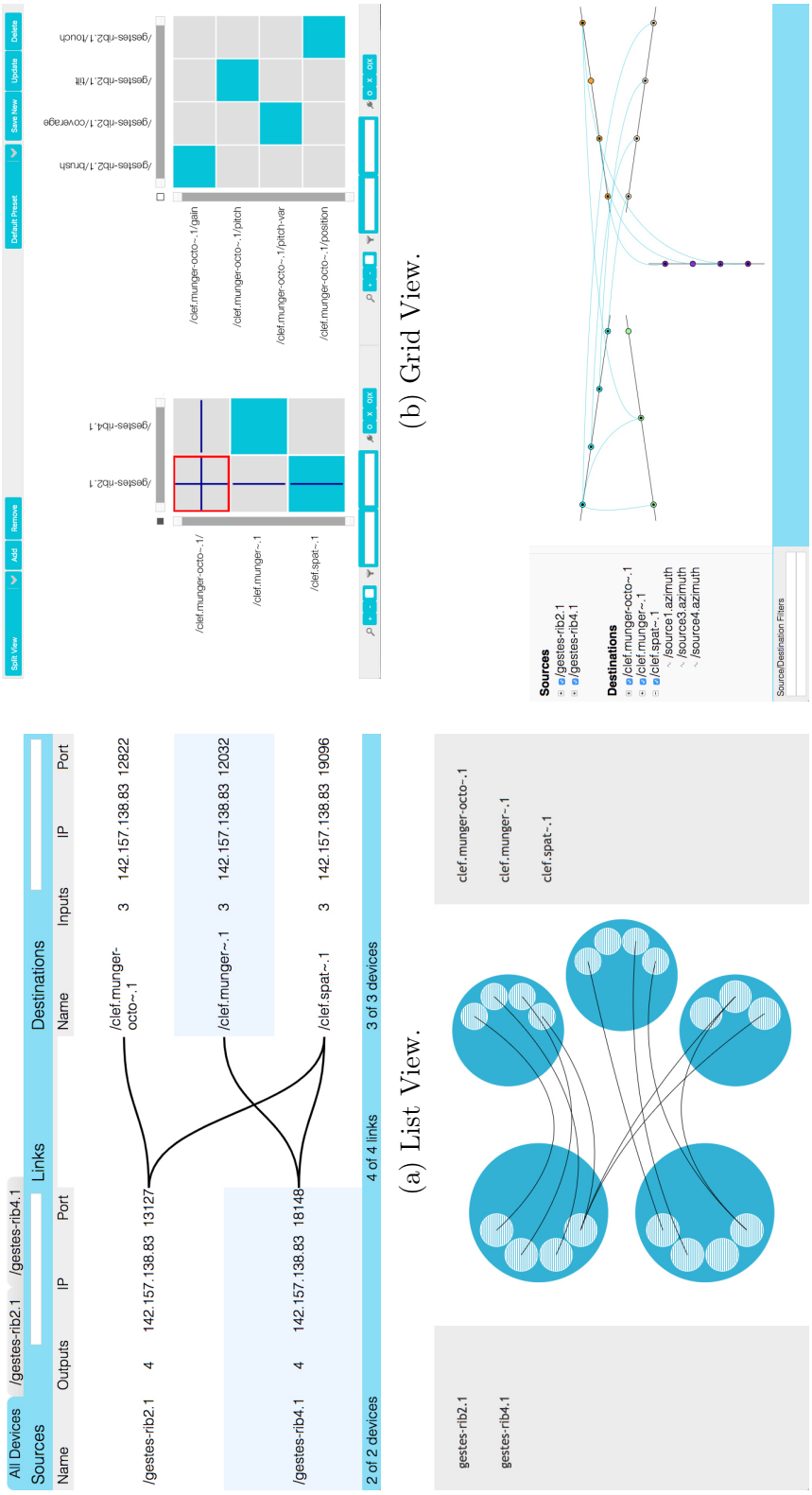
- *Semantic labelling of signals.* Signals can be defined according to human-readable names, and the signal characteristics are exposed as metadata for easy identification.

- *Signal processing and mapping within the connections.* A connection between two signals in libmapper can be subject to internal processing, basic mathematical operations like scaling, inversion, filtering, and combining for "many-to-one" (Hunt, Wanderley, and Paradis 2003) mappings.

- *Exposing mapping is a separate, modifiable entity.* This allows mappings to be visually represented and modified using internal tools, not necessarily subject to a single interaction paradigm.

As a distributed connectivity framework and software library, libmapper is intended to be incorporated into device drivers that emit information and synthesis applications that receive them. Once a device is implemented with libmapper interfaces, the signals will become available on the libmapper network and accessible via an "admin" bus. Tools that interact on this bus will then be able to initiate point to point connections and mappings between signals on the libmapper network.

Visual tools that access the "admin" bus allow interaction with all the devices and signals on the network, and webmapper (J. Wang et al. 2019) is one of these tools. Webmapper provides a browser based visual interface for interacting with libmapper devices, and presents multiple views of a mapping, as shown in 5.1.



(a) List View.

(b) Grid View.

(c) Balloon View.

(d) Hive View.

Fig. 5.1 Different visual representations provided by webmapper.

Going beyond the different visual representation of the mapping between devices and signals, we also explored the ability to keep track of the mapping structure as they evolve over time via a visual "Version Control" paradigm (J. Wang et al. 2017). This was identified as a useful feature when attempting to restore or evaluate previous configurations of a mapping during workshop settings.

In Figure 5.2 we present a prototype libmapper-based application that displays a "List" View (similar to webmapper), but with a user interface that allows states of the mapping to be saved, browsed, and restored. This feature was motivated through prior experience where exploration was hampered by the fear of losing prior states of an established mapping during the workshop setting, which was similarly identified by (Shneiderman et al. 2006).

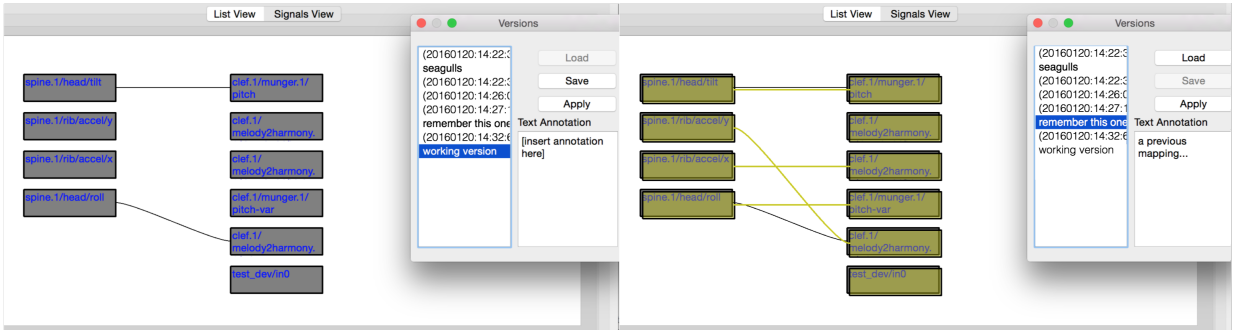


Fig. 5.2 Screenshot of the "Version Controlled" mapping editor showing working version (left), and the visual highlighting that reveals differences between the current and previous versions when interactively browsing through the history of previously stored versions.

The combination of these tools, supported by the underlying libmapper framework, provide new ways of interacting with devices and signals in the context of mapping design. webmapper has been used not only for the creation of mappings, but for research on how mappings are created (West et al. 2021).

5.2.1 Implication of Evaluation Findings on Mapping Tools

From the evaluations described in the earlier part of this thesis, we can identify some relationships between our findings and design considerations for mapping tools and frameworks.

First, we see that from a usability perspective, mapping frameworks such as libmapper can provide useful features by creating quick connections which can then better expose the features of flexible standards such as OSC without users having to keep track of explicit connection endpoints (such as the IP address and ports of receivers, and expected data ranges). To make use of the more advanced mapping features of libmapper, it would also be possible to build bridging end-points to bring more standardized devices (such as legacy MIDI controllers or synthesizers) into the mapping network. For platforms such as the ESP32 that support the networking stack already, it would be possible to embed the libmapper framework in the microcontroller firmware. There is on-going work to support libmapper-compatible sensor interfaces directly[5], so that embedded microcontroller-based devices can appear as standalone libmapper devices without the need for a host system running a bridging application.

In terms of visual mapping tools, we have described features of webmapper and the prototype versioning system that provide alternative visual representations of mappings, and the ability to iterate through previous mapping states in time. From the perspective of sensor interface performance (such as bandwidth and latency measurements), it would be interesting to be able to present the live sensor interface and network performance state into these tools. Such insights will afford users a better awareness of the real-time state of the devices and isolate potential performance issues. Additionally, the kind of evaluations we described in the earlier parts of the thesis can also be applied to verify any performance

[5] `https://github.com/mathiasbredholt/libmapper-arduino`

impacts when signals are routed through the libmapper network.

5.3 Application to the T-Stick

In this section we present how the findings presented in this thesis can be applied to an example device implementation developed at IDMIL from both performance and usability perspectives.

The author was involved in the redesign of the instrument and supported the construction of successive versions (Nieva et al. 2018). As the current version of the T-Stick makes use of an ESP32 microcontroller, it would be possible to implement either BLE or Wi-Fi based interfaces for communicating with a host system. Following is a presentation of how the findings from previous sections of the thesis can apply in the context of the T-Stick implementation.

5.3.1 Scaling Effects

The T-Stick sensor data includes Inertial Measurement Unit (IMU) (9×4-byte floats), pressure (1×4-byte float), and touch data (4 bytes, or more for longer versions of the instrument), and the slowest sensor on the device operates at 100 Hz. Based on this information, we can see that it would be difficult to scale beyond a few T-Sticks via BLE due to the 133 Hz maximum per-device message rate and 500 Hz total rate across up to five devices, as observed in Chapter 4.4. On the other hand, Wi-Fi appears to be sufficient for a number of concurrently operating devices on a single channel. However, as we discovered in Chapter 4.1, the latency increased as more devices were added.

While Wi-Fi has better observed scaling characteristics from both a per-device transmission rate and device multiplicity perspective compared to BLE, we do notice that the

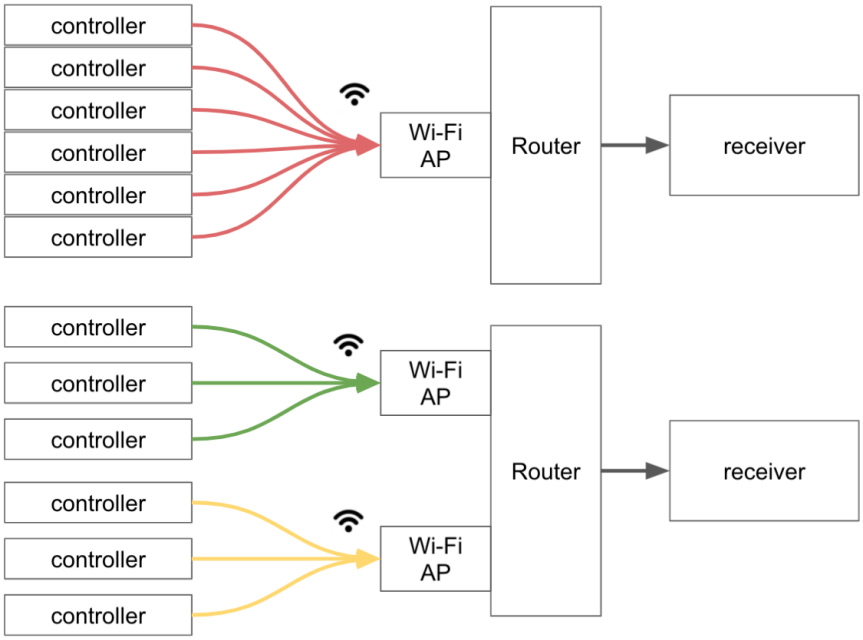


Fig. 5.3 Two Wi-Fi configurations showing six devices operating on a single channel versus two groups of three devices on two separate channels.

10 ms threshold is exceed between 6 and 7 concurrent devices operating at 100 Hz. However, an ensemble performance consisting of more than 7 wireless T-Sticks is still possible if either the sampling rate is reduced or additional wireless channels are employed. The former could be achieved either through the reduction of sampling rates or the *exclusive* use of lower bandwidth, high-level performance gestures (Meneses, Fukuda, and Wanderley 2020) that are less frequent in duration compared to the raw signals that are used to compute them. The latter configuration can be implemented using multiple wireless access points on different channels (or multi-radio access points) connected to a single network, as shown in Figure 5.3, which will increase the total network capacity. Since there will be additional networking hardware in between, the actual latency performance will need to be verified to see if the additional latency is acceptable. However, based on existing knowledge regarding wired Ethernet performance, this extra latency should be extremely low, and in

the sub-millisecond range (Loeser and Haertig 2004).

5.3.2 Usability and Connectivity

While Wi-Fi has an edge over BLE when it comes to raw performance, the potential for BLE-MIDI to operate seamlessly with existing synthesis applications is interesting for directly connecting to synthesis environments if on-board MIDI mappings, potentially connected to any high-level input gestures (Meneses, Fukuda, and Wanderley 2020) are implemented. This way, after pairing with the host device it would be possible to control any commercial synthesizer using the T-Stick as an alternate controller, similar to how existing commercially available keyboard or breath controlled MIDI input devices operate with synthesizers. On the other hand, the Wi-Fi implementation of the T-Stick could potentially be used with the ESP32 port of libmapper to make use of the auto discovery and mapping features, and allow the device to be used with the visual mapping tools presented in Section 5.2.

Chapter 6

Summary, Discussion, and Conclusion

In this concluding chapter, we first present a summary of the findings of the thesis that addressed the first three primary questions of minimal latency, maximum transmission rate, device multiplicity scaling and general compatibility concerns. We then present ways of extending the evaluations and potential future directions of the work, followed by some final concluding remarks.

6.1 Summary

In this thesis, we provided an empirical evaluation of wireless sensor interfaces, focusing on the performance and usability of protocols in the context of DMIs. We first presented the overall context of the work, as well as related work in existing sensor interface implementations and latency evaluations.

In terms of evaluations we first focused on the issue of quantifying latency of wireless interfaces by providing a methodology for, and carrying out, measurements on protocols including BLE, LoRa, and Wi-Fi on the ESP32 as well as legacy devices to be compared,

such as in the case of the industrial collaboration presented in Chapter 4.2. Results showed that a minimum end-to-end latency of around 7 ms is possible on Wi-Fi with a maximum transmission rate of around 2300 Hz, suggesting that the Wi-Fi protocol is the most suitable choice for most of wireless musical applications, provided that designers take into account its limitations with respect to the number of devices in use simultaneously, the maximum transmission rate, and background wireless traffic.

We then discussed usability aspects in terms of compatibility and connectivity, starting with how each tested protocol interoperates with existing platforms and environments. From there we presented several developments aimed to improve tools in the context of DMI design, focusing on the support for discovery, connectivity, and mapping. We concluded this part by presenting an application of the latency measurements and usability considerations to an existing instrument developed at IDMIL, the T-Stick, showing how our findings can be applied in a practical example.

6.2 Discussion and Future Directions

In this section we discuss limitations of the presented series of evaluations, as well as ways that the tests can be extended to include other relevant performance features regarding wireless interfaces in general.

6.2.1 Limitations of Presented Evaluations

While we have presented a number of tests that provide hardware interface / messaging protocol comparisons, transmission rate and device scaling, it would be possible to provide a more comprehensive coverage across these variables. While an exhaustive testing across all parameters of interest may be unpractical from a time perspective, the following

combinations may be of interest in the near future:

- *Scaling Parameters.* While we have experimented with a range of transmission rate and device scaling, a larger number of configurations could be tested, especially for Wi-Fi as the maximum available bandwidth is quite high. Additionally, a better quantification of the total bandwidth from a message payload perspective could be applied to provide a better understanding of the total carrying capacity of a network from a practical perspective.

- *Host Platform.* The evaluations were found for MacOS as the message receiver as well as synthesis host running Max/MSP. It would be interesting to see if there are performance differences in terms of message parsing and processing on other platforms as DMI implementation is not limited to the Apple desktop platform.

- *Investigation into BLE Connection Interval.* Since the maximum transmission rate as well as minimum latency is affected by this value and it is relatively high at 7.5 ms, it is worth investigating whether configurations can be changed on the Operating System (OS) and BLE stack level to yield better performance.

- *Routing Hardware Performance.* Since we made use of a Wi-Fi router for the tests, the performance of the particular routing hardware could also be compared with alternatives to see the impact that routers may have.

6.2.2 Generalized Hardware

In our work the majority of the wireless implementations were done on a single hardware platform. While this has the benefits of facilitating a direct comparisons of a wireless interface and messaging protocols, it limits our findings to the particular hardware in question,

the ESP32 microcontroller. Beyond testing with different hardware implementations, it would also be interesting to perform tests on commercially available devices such as the controllers presented in Appendix A. Strategies for triggering such devices, including the clap-board mechanism described in Chapter 4.5.2, could be used to non-invasively trigger a manufactured controller, for example.

6.2.3 Jitter, Time-Stamps, and Consistency

One relatively obvious metric that was revealed in our latency measurements was that most of the interface configurations exhibited a noticeable amount of jitter, that often increased as the latency values increased due to network traffic. This was evident by the trend of flatter ECDF curves as the average latencies increased. Like latency, it is not always clear what the acceptable amounts are for a given application, and thus far studies have been done on relatively limited contexts such as (Jack et al. 2018), enabled by platforms where the latency and jitter can be tightly controlled down to the individual sample level (McPherson 2017). With the availability of such tools, further studies with specific musical control contexts are possible and are interesting avenues of future research.

As mentioned previously, to counteract jitter it is always possible to add a variable delay up to the maximum expected latency value, at the cost of increasing the overall latency by this buffer length (Brandt and Dannenberg 1998). The BLE MIDI (MMA 2015) and OSC (Freed 1997) specifications both provide built-in and optional times-tamping capability to messages which can be employed to recover the exact timing behaviour of the messages. This is especially relevant for strict timing requirements when the messages are passed through a variable latency system such as a wireless channel.

Overall, the trade-offs between adding latency to eliminate jitter, the context-dependent requirements for jitter, and the actual amount of jitter present for a given transmission

channel are interdependent parameters that can be optimized for a particular application.

Finally, when it comes to transmission through a potentially unreliable medium, there is always the potential for extremely large or completely dropped packets. How the protocol handles transmission reliability. Dealing with timeouts, re-transmissions, and the general issue of Quality of Service (QoS) in the context of DMI applications is a general concern that is a part of the larger issue of reliable and consistent performance of these devices that can be addressed from both the protocol and application level.

6.2.4 Test Jig, Connection Intervals, and Synchronization

As described in Chapter 4.6, the implementation of the test jig should be done with care to avoid unintentional synchronization between the measurement device and the system under test, while understanding that this synchronization may not have an impact and also be useful under certain conditions. Perhaps another, potentially more important consequence of this finding beyond the design of the measurement jig itself is on how the sensor interface operates. The timing between successive sampling and transmission on the interface should not contain any fixed delay loops for rate control to avoid this synchronization issue. Instead, *time-elapsed* intervals can be used to control the actual transmission rate.

6.2.5 Power, Range, Reliability, and Other Parameters

While we have mentioned a basic measurement of power consumption as a function of transmission rates for each tested protocol combination in Chapter 4.3, this is an interesting area of investigation especially when portable battery powered devices are used. At the extreme ends of the spectrum, finger-worn controllers such as the Spectro Ring [1] or Genkii Wave [2]

[1] `https://sphero.com/collections/all/family_specdrums`
[2] `https://genkiinstruments.com/`

require extremely compact physical dimensions, and demand the use of small batteries.

Beyond the latency and power performance, another important metric for wireless device performance is range, and its subsequent effect on reliability as the distance between transmitter and receiver increases. While not as crucial for tabletop use as the case of a controller connected to a computer nearby, the range requirements can be greater for ensemble and stage performance settings.

6.2.6 Additional RF Systems and Topologies

In our work we have primarily focused on RF systems with native compatibility on commodity systems (with the exception of LoRa for the one investigation described in Chapter 4.3). One immediate configuration is the newer version of Wi-Fi operating on the 5Ghz band that has more bandwidth and less interference (Gast 2005). While it is readily available on some embedded platforms such as the current generation Raspberry Pi and various mobile devices, it has yet to become accessible on microcontroller based devices. Even more recent implementations such as 802.11ax (or Wi-Fi 6) (Bellalta 2016), as well as other potentially interesting emerging technologies include Bluetooth Mesh (Baert et al. 2018) and 5G cellular (Turchet et al. 2018) are also of interest for future evaluation as they become available.

Perhaps of further interest, is how these different networking solutions could offer better connection topologies that could be more fitting for a particular application. In our investigations we have primarily dealt with the minimal latency measurements associated with a sender transmitting to a single receiver, and the scaling of devices assume a single centralized synthesis system that processes all the data. While this is sufficient for obtaining the performance of this connection topology, a different application may require a different connection structure. Some examples of this is a one-to-many situation as presented in

(D'Alessandro et al. 2012; Hattwick, Franco, and Wanderley 2017) where a single central coordinator transmits control messages to a number of receivers that synthesize output individually. Such a configuration would then be more suited to a broadcast configuration.

6.2.7 Towards an Evaluation Framework for Sensor Interfaces

One final, perhaps indirect contribution of this thesis is the establishment and on-going development of the set of open-source tools for evaluations. Building upon the work first presented by (McPherson, Jack, and Moro 2016), we hope to collaborate in building a community of users around an accessible set of tools to verify the performance of sensor interfaces and protocols used by the DMI community. All the source code, hardware schematics, and circuit diagrams have been made available on a public repository[3]. One potential avenue to publish a centralized database of findings is a public forum such as SensorWiki (Wanderley et al. 2006) where the evaluation results can be disseminated across the global community, in addition to the traditional avenues of conference and journal publications.

6.3 Conclusion

In this thesis, we have presented the motivation and evaluation of wireless sensor interfaces in the context of DMI design. Having presented the initial context and motivation for the work, we described our evaluation system which was constructed from a user application perspective that provides empirical measurements of protocol performance that resembles specific deployment settings (i.e. using available software and hardware found on commodity computational devices). While there were specific constraints on the choice of hardware

[3] `https://github.com/IDMIL/interface-tests`

and protocols, they also resemble what would be realistic choices encountered from the perspective of a DMI designer who is not necessarily interested in low-level implementation details of custom RF hardware. At the same time, the presented system of evaluation can be used for any system that contains the basic features of converting an input signal to output, and we have demonstrated ways of adapting the system to work with audio-to-audio, and in conjunction with other media systems such as a motion-capture device.

In terms of measured results, we clearly see a performance advantage of Wi-Fi based interfaces that is consistent with prior research in the literature showing its relatively good latency (McPherson, Jack, and Moro 2016) and bandwidth (Mitchell et al. 2014) performance. However, we also acknowledge that in a potentially unreliable transmission medium, there is always the chance of dropped packets due to network congestion or interference, and such wireless channels are not always controllable in a performance setting.

Overall, despite the limits of wireless interfaces from a performance perspective as revealed by the various evaluations presented in this thesis, its convenience and widespread availability make it usable in DMI applications provided that the limitations are well understood. As new and improved wireless technologies are released, we hope that the increased adoption of the testing methodology presented in this thesis can be adapted towards producing more readily available evaluation data on emerging devices.

Bibliography

Adelantado, Ferran, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne (2017). "Understanding the Limits of LoRaWAN". In: *IEEE Communications Magazine* 55.9, pp. 34–40.

Atzori, Luigi, Antonio Iera, and Giacomo Morabito (2010). "The Internet of Things: A Survey". In: *Computer Networks* 54.15, pp. 2787–2805.

Augustin, Aloÿs, Jiazi Yi, Thomas Clausen, and William Mark Townsley (2016). "A study of LoRa: Long Range & Low Power Networks for the Internet of Things". In: *Sensors* 16.9, p. 1466.

Baalman, Marije A. J., Vincent De Belleval, Joseph Malloch, Joseph Thibodeau, C. Salter, and Marcelo M. Wanderley (2010). "Sense/Stage-Low Cost, Open Source Wireless Sensor Infrastructure For Live Performance And Interactive, Real-Time Environments." In: *International Computer Music Conference*. New York, USA.

Baert, Mathias, Jen Rossey, Adnan Shahid, and Jeroen Hoebeke (2018). "The Bluetooth Mesh standard: An Overview and Experimental Evaluation". In: *Sensors* 18.8, p. 2409.

Bellalta, Boris (2016). "IEEE 802.11 ax: High-efficiency WLANs". In: *IEEE Wireless Communications* 23.1, pp. 38–46.

Berdahl, Edgar and Wendy Ju (2011). "Satellite CCRMA: A Musical Interaction and Sound Synthesis Platform". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Oslo, Norway, pp. 173–178.

Bevilacqua, Frédéric, Rémy Müller, and Norbert Schnell (2005). "MnM: a Max / MSP Mapping Toolbox". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Vancouver, Canada.

Bhagwat, Pravin (2001). "Bluetooth: Technology for Short-range Wireless Apps". In: *IEEE Internet Computing* 5.3, pp. 96–103.

Bi, Qi, G. L. Zysman, and Hank Menkes (2001). "Wireless Mobile Communications at the Start of the 21st Century". In: *IEEE Communications Magazine* 39.1, pp. 110–116.

Birnbaum, David, Rebecca Fiebrink, Joseph Malloch, and Marcelo M. Wanderley (2005). "Towards a Dimension Space for Musical Devices". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Vancouver, Canada, pp. 192–195.

Bongers, Bert (2000). "Physical Interfaces in the Electronic Arts Interaction Theory and Interfacing Techniques for Real-time Performance". In: *Trends in Gestural Control of Music* 2000, pp. 41–70.

Brandt, Eli and Roger B. Dannenberg (1998). "Low-latency Music Software using Off-the-shelf Operating Systems". In: *Proceedings of the International Computer Music Conference*. Ann Arbor, USA: Carnegie Mellon University.

Bulow, Jeremy (1986). "An Economic Theory of Planned Obsolescence". In: *The Quarterly Journal of Economics* 101.4, pp. 729–749.

Buxton, Bill (1997). "Artists and the Art of the Luthier". In: *ACM SIGGRAPH Computer Graphics*. Vol. 31. 1. ACM, pp. 10–11.

Calegario, Filipe, Marcelo M. Wanderley, Stéphane Huot, Giordano Cabral, and Geber Ramalho (2017). "A Method and Toolkit for Digital Musical Instruments: Generating Ideas and Prototypes". In: *IEEE MultiMedia* 24.1, pp. 63–71.

Candy, Linda (2006). "Practice based research: A guide". In: *CCS report* 1, pp. 1–19.

Card, Stuart, Jock Mackinlay, and George Robertson (1991). "A Morphological Analysis of the Design Space of Input Devices". In: *ACM Transactions on Information Systems* 9.2, pp. 99–122. ISSN: 10468188. DOI: 10.1145/123078.128726.

Carôt, Alexander and Christian Werner (2009). "Fundamentals and Principles of Musical Telepresence". In: *Journal of Science and Technology of the Arts* 1.1, pp. 26–37.

Coduys, Thierry, Cyrille Henry, and Arshia Cont (2004). "TOASTER and KROONDE: High-Resolution and High-Speed Real-time Sensor Interfaces." In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Hamamatsu, Japan, pp. 205–206.

Cook, Perry R (2001). "Principles for Designing Computer Music Controllers". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Seattle, USA, pp. 1–4. DOI: 10.1101/gr.075622.107.

— (2009). "Re-Designing Principles for Computer Music Controllers: A Case Study of SqueezeVox Maggie". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Vol. 11. Pittsburgh, USA, pp. 218–221.

D'Alessandro, Nicolas, Aura Pon, Johnty Wang, David Eagle, and Sidney Fels (2012). "A Digital Mobile Choir : Joining Two Interfaces towards Composing and Performing Collaborative Mobile Music". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Ann Arbor, USA.

Ferguson, Sean and Marcelo M. Wanderley (2010). "The McGill Digital Orchestra: Interdisciplinarity in Digital Musical Instrument Design". In: *Journal of Interdisciplinary Music Studies* 4.2, pp. 17–35.

Fiebrink, Rebecca, Dan Trueman, and Perry R Cook (2009). "A Metainstrument for Interactive, On-the-fly Machine Learning". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Pittsburgh, USA, pp. 280–285.

Fléty, Emmanuel (2005). "The Wise Box: a Multi-performer Wireless Sensor Interface using WiFi and OSC". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Vancouver, Canada, pp. 266–267.

Fléty, Emmanuel and Côme Maestracci (2011). "Latency Improvement in Sensor Wireless Transmission Using IEEE 802.15.4". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. June. Oslo, Norway, pp. 409–412.

Fraden, Jacob (2010). *Handbook of Modern Sensors*. Vol. 3. Springer.

Franco, Ivan and Marcelo M. Wanderley (2016a). "Prynth: A framework for Self-contained Digital Music Instruments". In: *International Symposium on Computer Music Multidisciplinary Research*. Springer. São Paulo, Brazil, pp. 357–370.

— (2016b). "The Mitt: Case Study in the Design of a Self-contained Digital Music Instrument". In: *International Symposium on Computer Music Multidisciplinary Research*. São Paulo, Brazil.

Freed, Adrian (1997). "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers". In: *International Computer Music Conference*. Thessaloniki, Greece.

Freed, Adrian, Amar Chaudhary, and Brian Davila (1997). "Operating Systems Latency Measurement and Analysis for Sound Synthesis and Processing Applications". In: *Proceedings of the International Computer Music Conference*. Thessaloniki, Greece.

Fukuda, Takuto, Eduardo A. L. Meneses, Travis West, and Marcelo M. Wanderley (2021). "The T-Stick Music Creation Project: An Approach to Building a Creative Community Around a DMI". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Shanghai, China.

Gast, Matthew (2005). *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, Inc.

Gomez, Carles, Joaquim Oller, and Josep Paradells (2012). "Overview and Evaluation of Bluetooth Low Energy: An emerging Low-power Wireless Technology". In: *Sensors* 12.9, pp. 11734–11753.

Harlow, Randall (2018). *Hyperorgan Mediation Technology*. Orgelpark Research Reports, Vol 5/2.

Hattwick, Ian, Ivan Franco, and Marcelo M. Wanderley (2017). "The Vibropixels: A Scalable Wireless Tactile Display System". In: *Human Computer International Conference*. Vancouver, Canada.

Hunt, Andy and Marcelo M. Wanderley (2002). "Mapping performer parameters to synthesis engines". In: *Organised Sound* 7.02, pp. 97–108. ISSN: 1355-7718. DOI: 10.1017/S1355771802002030.

Hunt, Andy, Marcelo M. Wanderley, and Matthew Paradis (2003). "The Importance of Parameter Mapping in Electronic Instrument Design". In: *Journal of New Music Research* 32.4, pp. 429–440. ISSN: 0929-8215. DOI: 10.1076/jnmr.32.4.429.18853.

Igoudin, Alex Lane (1997). "Impact of MIDI on electroacoustic Art Music". In: *Proceedings of the International Computer Music Conference*. Thessaloniki, Greece: Stanford University.

Jack, Robert H., Adib Mehrabi, Tony Stockman, and Andrew McPherson (2018). "Action-sound Latency and the Perceived Quality of Digital Musical Instruments: Comparing Professional Percussionists and Amateur Musicians". In: *Music Perception: An Interdisciplinary Journal* 36.1, pp. 109–128.

Jiang, Xiaolin, Hossein Shokri-Ghadikolaei, Gabor Fodor, Eytan Modiano, Zhibo Pang, Michele Zorzi, and Carlo Fischione (2018). "Low-latency networking: Where latency lurks and How to Tame it". In: *Proceedings of the IEEE* 107.2, pp. 280–306.

Kitchen, La (2016). *Kroonde Gamma User Manual*. English. Version 1.6. La Kitchen. December, 2016.

Kostek, B. and A. Czyżewski (1993). "Investigation of Articulation Features in Organ Pipe Sound". In: *Archives of Acoustics* 18.3. ISSN: 2300-262X.

Kushner, David (2011). "The Making of Arduino". In: *IEEE spectrum* 26.

Lehrman, Paul D (2020). "MIDI 2.0: Promises and Challenges". In: *Music Encoding Conference 2020*. Medford, USA.

Lehrman, Paul D and Tim Tully (1993). *MIDI for the Professional: The Essential References for the Serious MIDI User. A Complete Technical Guide to Today's MIDI Devices and Applications Expert Advice and Creative Techniques to Help You Get the Most Out of Your MIDI Setup*. Amsco.

Lester, Michael and Jon Boley (2007). "The Effects of Latency on Live Sound Monitoring". In: *Audio Engineering Society Convention 123*. Audio Engineering Society. New York, USA.

Ljungström, Andreas and Jack Panikian (2016). "Implementation and Evaluation of Bluetooth Low Energy for Musical Devices". Bachelor's Thesis. KTH Royal Institute of Technology.

Loeser, Jork and Hermann Haertig (2004). "Low-latency Hard Real-time Communication over Switched Ethernet". In: *Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004.* IEEE. Catania, Italy, pp. 13–22.

Loy, Gareth (1985). "Musicians Make a Standard: the MIDI Phenomenon". In: *Computer Music Journal* 9.4, pp. 8–26.

Machover, Tod (1992). *Hyperinstruments: A Progress Report, 1987-1991*. MIT Media Laboratory.

MacMillan, K., M. Droettboom, and I. Fujinaga (2001). "Audio Latency Measurements of Desktop Operating Systems". In: *Proceedings of the International Computer Music Conference*. Havana, Cuba, pp. 259–262.

Madgewick, Sebastian and Thomas Mitchell (2013). "x-OSC - A Versatile Wireless I/O Device for Creative Music Applications". In: *Proceedings of the Sound and Music Computing Conference*. Stockholm, Sweden.

Maier, Martin, Mahfuzulhoq Chowdhury, Bhaskar Prasad Rimal, and Dung Pham Van (2016). "The Tactile Internet: Vision, Recent Progress, and Open Challenges". In: *IEEE Communications Magazine* 54.5, pp. 138–145.

Mäki-Patola, Teemu and Perttu Hämäläinen (2004). "Latency Tolerance for Gesture Controlled Continuous Sound Instrument Without Tactile Feedback". In: *Proceedings of the International Computer Music Conference*. Miami, USA.

Malloch, Joseph, Stephen Sinclair, and Marcelo M. Wanderley (2014). "Distributed tools for interactive design of heterogeneous signal networks". In: *Multimedia Tools and Applications*, pp. 1–25. ISSN: 13807501. DOI: 10.1007/s11042-014-1878-5.

Malloch, Joseph and Marcelo M. Wanderley (2007). "The T-Stick : From Musical Interface to Musical Instrument". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. New York, USA, pp. 66–69. DOI: 10.1145/1279740.1279751.

McPherson, Andrew (2017). "Bela: An Embedded Platform for Low-Latency Feedback Control of Sound". In: *The Journal of the Acoustical Society of America* 141.5, pp. 3618–3618.

McPherson, Andrew, Robert H. Jack, and Giulio Moro (2016). "Action-Sound Latency: Are Our Tools Fast Enough?" In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Brisbane, Australia. ISBN: 978-1-925455-13-7.

McPherson, Andrew and Victor Zappi (2015). "An Environment for Submillisecond-Latency Audio and Sensor Processing on BeagleBone Black". In: *Audio Engineering Society Convention 138*. Audio Engineering Society. Warsaw, Poland.

Meneses, Eduardo A. L., Sérgio Freire, and Marcelo M. Wanderley (2018). "GuitarAMI and GuiaRT: Two Independent yet Complementary Augmented Nylon Guitar Projects". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Blacksburg, USA.

Meneses, Eduardo A. L., Takuto Fukuda, and Marcelo M. Wanderley (2020). "Expanding and Embedding a High-Level Gesture Vocabulary for Digital and Augmented Musical Instruments". In: *International Conference on Human-Computer Interaction*. Springer. Copenhagen, Denmark, pp. 375–384.

Meneses, Eduardo A. L., Johnty Wang, Sergio Freire, and Marcelo M. Wanderley (2019). "A Comparison of Open-Source Linux Frameworks for an Augmented Musical Instrument Implementation". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Porto Alegre, Brazil.

Miranda, Eduardo Reck and Marcelo M. Wanderley (2006). *New Digital Musical Instruments: Control and Interaction Beyond the Keyboard*. Vol. 21. AR Editions, Inc.

Mitchell, Thomas, Sebastian Madgwick, Simon Rankine, Geoffrey Hilton, Adrian Freed, and Andrew Nix (2014). "Making the Most of Wi-Fi: Optimisations for Robust Wireless Live Music Performance". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. London, UK.

MMA, MIDI (2015). *Specification for MIDI over Bluetooth Low Energy (BLE-MIDI) 1.0*. Tech. rep. The MIDI Manufacturers Association.

Moog, Robert A (1986). "MIDI: Musical Instrument Digital Interface". In: *Journal of the Audio Engineering Society* 34.5, pp. 394–404.

Moore, F Richard (1988). "The Dysfunctions of MIDI". In: *Computer music journal* 12.1, pp. 19–28.

Morreale, Fabio et al. (2017). "Design for longevity: Ongoing use of instruments from NIME 2010-14". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Aalborg, Denmark.

Mulder, Axel (1995). "The I-Cube System: Moving Towards Sensor Technology for Artists". In: *Proceedings of the Sixth Symposium on Electronic Arts (ISEA 95)*. Montreal, Canada.

Nelson, Mark and Belinda Thom (2004). "A Survey of Real-Time MIDI Performance". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Hamamatsu, Japan, pp. 35–38. DOI: 10.5281/zenodo.1176643.

Nielsen, Jakob (1994). *Usability Engineering*. Morgan Kaufmann.

Nieva, Alex, Johnty Wang, Joseph Malloch, and Marcelo M. Wanderley (2018). "The T-Stick: Maintaining a 12 year-old Digital Musical Instrument". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Blacksburg, USA.

Oliveros, Pauline, Sarah Weaver, Mark Dresser, Jefferson Pitcher, Jonas Braasch, and Chris Chafe (2009). "Telematic music: six perspectives". In: *Leonardo Music Journal* 19.1, pp. 95–96.

Palshikar, Girish et al. (2009). "Simple Algorithms for Peak Detection in Time-Series". In: *Proceedings of the 1st International Conference on Advanced Data Analysis, Business Analytics and Intelligence*. Vol. 122. Ahmedabad, India.

Pennycook, Bruce W. (1985). "Computer-music Interfaces: a Survey". In: *ACM Computing Surveys* 17.2, pp. 267–289. ISSN: 03600300. DOI: 10.1145/4468.4470.

Rottondi, Cristina, Chris Chafe, Claudio Allocchio, and Augusto Sarti (2016). "An Overview on Networked Music Performance Technologies". In: *IEEE Access* 4, pp. 8823–8843.

Rovan, Joseph Butch, Marcelo M. Wanderley, Shlomo Dubnov, and Philippe Depalle (1997). "Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance". In: *Kansei, The Technology of Emotion. Proceedings of the AIMI International Workshop*. Genova, Italy, pp. 68–73.

Rowe, Robert (1992). *Interactive Music Systems: Machine Listening and Composing*. MIT press.

— (2005). "Personal Effects: Weaning Interactive Systems from MIDI". In: *Proceedings of the Spark Festival*.

Santos, Geise, Johnty Wang, Carolina Medeiros, Marcelo M. Wanderley, Tiago Tavares, and Anderson Rocha (2021). "Latency Analysis of a Hybrid Setup Comprising an Optical Motion Capture System and Wireless Sensor Interface". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Shanghai, China.

Scavone, Gary P and Perry R Cook (2004). "RtMidi, RtAudio, and a Synthesis Toolkit (STK) Update". In: *Proceedings of the International Computer Music Conference*. Miami, USA.

Severance, Charles (2013). "Eben Upton: Raspberry pi". In: *IEEE Computer* 46.10, pp. 14–16.

Shneiderman, Ben, Pamela Jennings, Gerhard Fischer, Brad Myers, Ernest Edmonds, and Mike Eisenberg (2006). "Creativity Support Tools: Report From a U.S. National Science Foundation Sponsored Workshop". In: *International Journal of Human-Computer Interaction* 20.2, pp. 61–77. DOI: 10.1145/1323688.1323689.

Steiner, Hans-Christoph (2006). "Towards a Catalog and Software Library of Mapping Methods". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. NIME '06. Paris, France: IRCAM - Centre Pompidou, pp. 106–109. ISBN: 2-84426-314-3.

Stevens, W. Richard (Jan. 1997). *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*. RFC 2001. RFC Editor. URL: http://www.rfc-editor.org/rfc/rfc2001.txt.

Stewart, Andrew D. (2009). "Digital Musical Instrument Composition: Limits and Constraints". In: *Proceedings of the Electroacoustic Music Studies Network Conference*. Buenos Aires, Argentina, pp. 3–8.

Tanaka, Atau and Bert Bongers (2002). "Global String: A Musical Instrument for Hybrid Space". In: *International Computer Music Conference*. Gothenburg, Sweden.

Texas Instruments (Oct. 2011). *INA219 Zero-Drift, Bidirectional Current/Power Monitor With I2C Interface*. INA219. Rev. G.

— (Oct. 2020). *LM393B, LM2903B, LM193, LM293, LM393 and LM2903 Dual Comparators*. LM393. Rev. AD.

Turchet, Luca, Carlo Fischione, Georg Essl, Damián Keller, and Mathieu Barthet (2018). "Internet of Musical Things: Vision and Challenges". In: *IEEE Access* 6.

Wanderley, Marcelo M. (Aug. 2002). "Mapping Strategies for Real-time Computer Music. Special Issue." In: *Organized Sound* 7.2.

Wanderley, Marcelo M., David Birnbaum, Joseph Malloch, Elliot Sinyor, and Julien Boissinot (2006). "Sensorwiki.org: A Collaborative Resource for Researchers and Interface Designers". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Paris, France, pp. 180–183.

Wanderley, Marcelo M. and Joseph Malloch (Mar. 2014). "Advances in the Design of Mapping for Computer Music. Special Issue." In: *Computer Music Journal*.

Wanderley, Marcelo M. and Nicola Orio (2002). "Evaluation of Input Devices for Musical Expression : Borrowing Tools from HCI". In: *Computer Music Journal* 26.3, pp. 62–76.

Wang, Johnty, Joseph Malloch, Fanny Chevalier, and Marcelo M. Wanderley (2017). "Versioning and Annotation Support for Collaborative Mapping Design". In: *Proceedings of the Sound and Music Computing Conference*. Espoo, Finland.

Wang, Johnty, Joseph Malloch, Stephen Sinclair, Jonathan Wilansky, Aaron Krajeski, and Marcelo M. Wanderley (2019). "Webmapper: A Tool for Visualizing and Manipulating Mappings in Digital Musical Instruments". In: *International Symposium on Computer Music Multidisciplinary Research*. Marseille, France, p. 823.

Wang, Johnty, Eduardo A. L. Meneses, and Marcelo M. Wanderley (2020). "The Scalability of WiFi for Mobile Embedded Sensor Interfaces". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Birmingham, UK.

Wang, Johnty, Axel Mulder, and Marcelo M. Wanderley (2019). "Practical Considerations for MIDI over Bluetooth Low Energy as a Wireless Interface". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Porto Alegre, Brazil.

Wang, Yonghao (2018). "Low Latency Audio Processing". PhD thesis. Queen Mary University of London.

Wessel, David and Matthew Wright (2002). "Problems and Prospects for Intimate Musical Control of Computers". In: *Computer Music journal* 26.3, pp. 11–14. ISSN: 0148-9267. DOI: 10.1162/014892602320582945.

West, Travis, Baptiste Caramiaux, Stéphane Huot, and Marcelo M. Wanderley (2021). "Making Mappings: Design Criteria for Live Performance". In: *Proceedings of the Conference on New Interfaces for Musical Expression*. Shanghai, China.

Wright, James L. and Eli Brandt (2001). "System-Level MIDI Performance Testing". In: *Proceedings of the International Computer Music Conference*. Havana, Cuba.

Wright, Matthew (2005). "Open Sound Control: An Enabling Technology for Musical Networking". In: *Organised Sound* 10.03, pp. 193–200.

Wright, Matthew, Ryan J. Cassidy, and Michael F Zbyszy (2004). "Audio and Gesture Latency Measurements on Linux and OSX Introduction and Prior Work". In: *Proceedings of the International Computer Music Conference*. Miami, USA, pp. 423–429.

Wu, Weixin (2011). "Measuring Digital System Latency from Sensing to Actuation at Continuous 1 Millisecond Resolution". PhD thesis. Clemson University.

Zimmermann, Hubert (1980). "OSI Reference Model: The ISO Model of Architecture for Open Systems Interconnection". In: *IEEE Transactions on communications* 28.4, pp. 425–432.

Appendices

Appendix A

Examples of Commercially Available BLE-MIDI Devices

Manufacturer	Model	Device Type	Details
ACPAD	ACPAD	Guitar Controller	Buttons attached to guitar
Artiphon	Orba	Touch / tap Controller	Touch and motion
Artiphon	INSTRUMENT 1	Guitar-like Controller	Strumming, fretting, pressing
CME	X-Key Air	Keyboard Controller	MIDI Keyboard
CME	WIDI Bud	USB to BLE-MIDI Adapter	Conversion adapter
Genki	Wave	Fingertip Motion	Motion and touch
Isla	KordBot	Chord Controller	Knobs, buttons, sliders
Livid	Minim	Control Interface	Knobs, buttons, sliders
Korg	micro/nanoKey	Keyboard Controller	MIDI Keyboard
Quicco	mi.1	MIDI to BLE-MIDI	Conversion adapter
Roland	Aerophone Series	Wind Controller	Wind controller and synthesizer
Roland	WM-01	BLE Adapter	Conversion adapter
ROLI	Blocks	Control interfaces	Reconfigurable control surfaces
ROLI	Seaboard	Keyboard Controller	Multi touch membrane keyboard
Sensel	Morph	Reconfigurable Interface	Multi-touch membrane surfaces
Sphero	Specdrums	Percussion Controller	Colour sensing trigger ring
Yamaha	SHS-500	Keytar Controller	BLE and USB MIDI Controller
Yamaha	MD-BT01	MIDI to BLE-MIDI	Conversion adapter
Yamaha	UD-BT01	USB-Host to BLE-MIDI	Conversion adapter
Zivix	jamstik	Guitar Controller	Guitar-like controller

Appendix B

Latency Hardware

B.1 Latency Jig Schematic

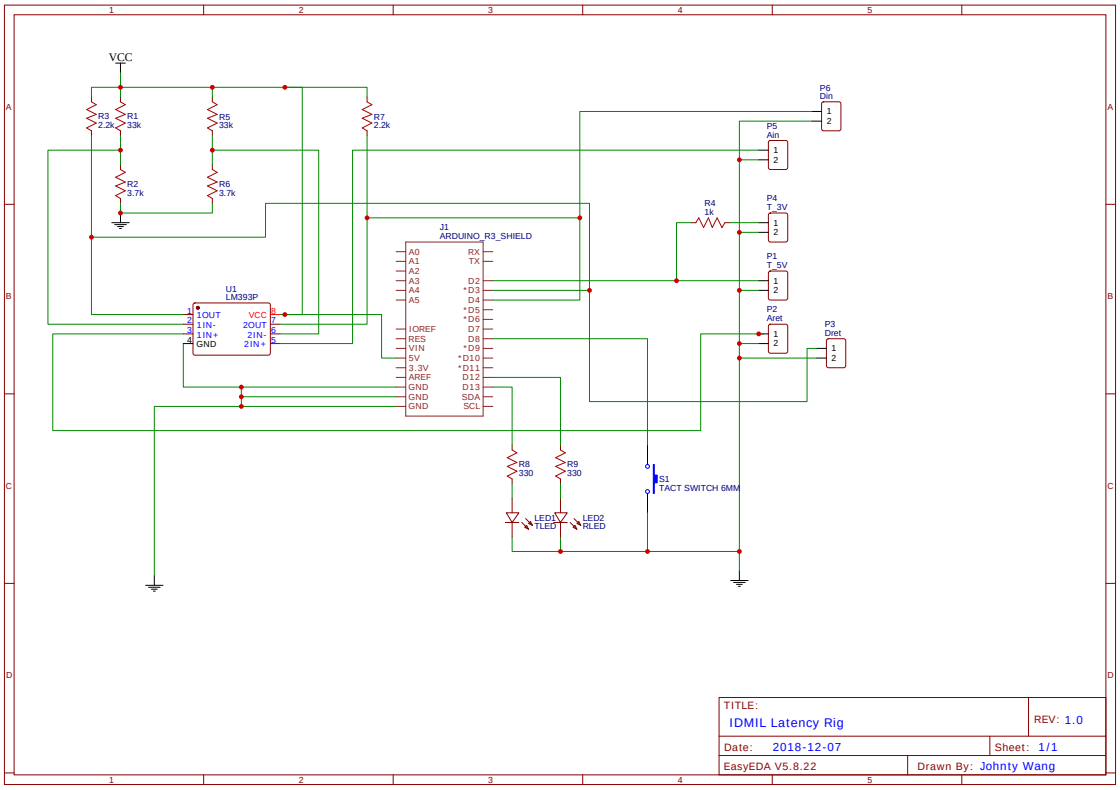


Fig. B.1 The schematic of the latency jig. Both channels of the comparator are connected for future expansion, but only a single channel is used for the evaluations described in this thesis.

B.2 Latency Jig PCB Layout

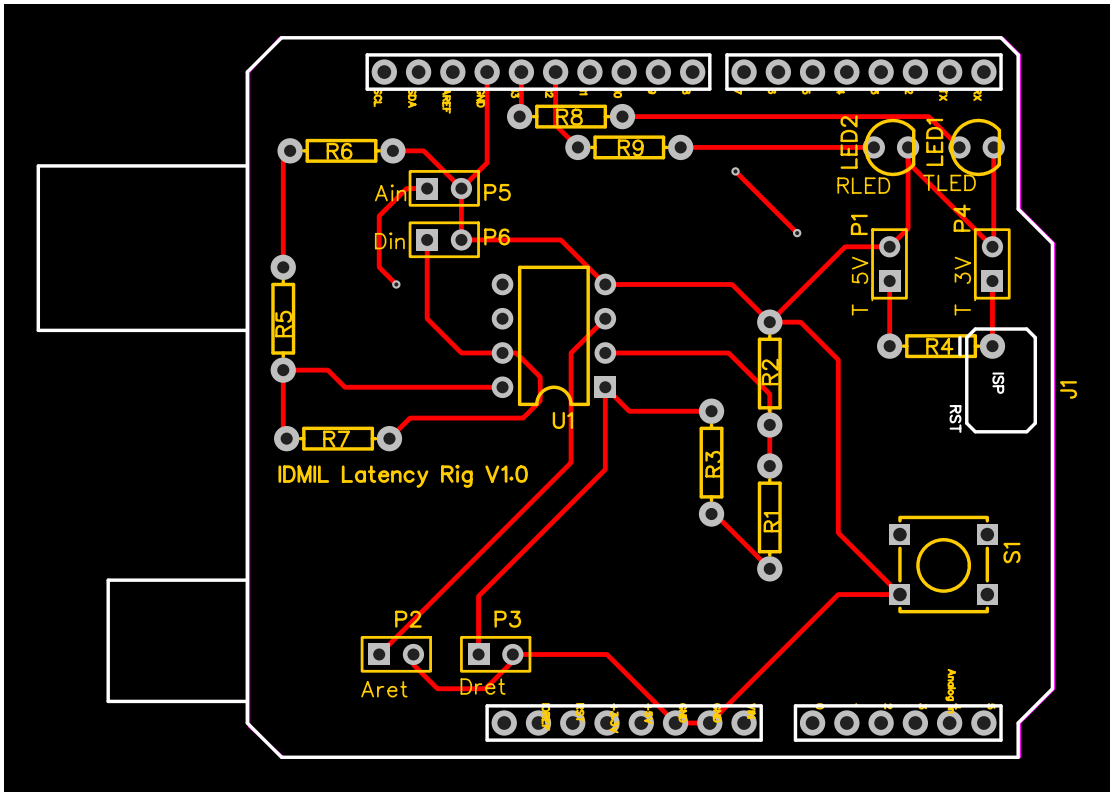


Fig. B.2 The PCB board layout of the latency jig Arduino "Shield". It is a simple two-layer board that can be cheaply manufactured by most PCB board houses, and the components are large for placement and soldering by people with novice electronics skills.

B.3 Latency Jig Bill of Materials

Table B.1 The Bill of Materials for the latency measurement jig, which contains a single comparator IC along with supporting discrete components of resistors, capacitors, and LEDs.

ID	Name	Designator	Footprint	Quantity
1	ARDUINO_R3_SHIELD	J1	UNO_R3_SHIELD	1
2	TLED	LED1	LED-3MM/2.54	1
3	RLED	LED2	LED-3MM/2.54	1
4	T_5V	P1	HDR-1X2/2.54	1
5	Aret	P2	HDR-1X2/2.54	1
6	Dret	P3	HDR-1X2/2.54	1
7	T_3V	P4	HDR-1X2/2.54	1
8	Ain	P5	HDR-1X2/2.54	1
9	Din	P6	HDR-1X2/2.54	1
10	33k	R1,R5	AXIAL-0.3	2
11	3.7k	R2,R6	AXIAL-0.3	2
12	2.2k	R3,R7	AXIAL-0.3	2
13	1k	R4	AXIAL-0.3	1
14	330	R8,R9	AXIAL-0.3	2
15	TACT SWITCH 6MM	S1	TACT6MM	1
16	LM393P	U1	DIP-8	1