Interactivity by Design: Interactive Art Systems Through Network Programming

by

Steven A. Bjornson
B.A., University of Victoria, 2012

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in Interdisciplinary Studies
in the areas of Computer Science and Visual Arts

Interactivity by Design: Interactive Art Systems Through Network Programming

by

Steven A. Bjornson
B.A., University of Victoria, 2012

Supervisory Committee

Dr. George Tzanetakis, Co-Supervisor
(Department of Computer Science)

Paul Walde, Co-Supervisor
(Department of Visual Arts)

Supervisory Committee

Dr. George Tzanetakis, Co-Supervisor
(Department of Computer Science)

Paul Walde, Co-Supervisor
(Department of Visual Arts)

ABSTRACT

Interactive digital art installations are fundamentally enabled by hardware and software. Through a combination of these elements an interactive experience is constructed. The first half of this thesis discusses the technical complexity associated with design and implementation of digital interactive installation. A system, *dreamIO*, is proposed for mediating this complexity through providing wireless building blocks for creating interactive installations. The technical details–both hardware and software–of this system are outlined. Measurements of the system are presented followed by analysis and discussion of the real world impact of this data. Finally, a discussion of future improvements is presented.

The second half of this thesis examines an example interactive installation, *Transcode*, which uses the proposed system as the building block for the piece. The piece is presented as evidence for the value of the proposed system and as a work of art in it's own right. The use of the *dreamIO* system is detailed followed by a discussion of the interactivity and aesthetic form of the work. The purposes of these specific design choices are then presented. Finally, the work is analyzed through a combination of Relational Aesthetics and Cybernetics.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Stephen Harrison, for teaching me everything I really needed to know.

Dr. George Tzanetakis and Paul Walde, for taking a chance on new territory.

the BC Arts Council, for funding me with a Scholarship.

Studio Robazzo, for giving me so much time on their laser cutter.

the Commune, for providing a fertile ground for learning, exploration, and growth.

and, El Gato Rojo, for everything.

"Neither the artist nor the mathematician may be able to tell you what constitutes the difference between a significant piece of work and an inflated trifle; but if he is not able to recognize this in his own heart, he is no artist and no mathematician."
Norbert Wiener

"And there still remained, for all [people] to share, the linked worlds of love and art. Linked, because love without art is merely the slaking of desire, and art cannot be enjoyed unless it is approached with love."
Arthur C. Clarke

DEDICATION

For my Parents for their love and support.

Chapter 1

Introduction

Through a combined study of computer science and visual art, I have spent time exploring the relationship between the ideas and technologies that underly digital computation and how these tools and the understanding of the digital domain can be used to generate novel artistic output. This study has yielded several pieces–see A.5 for details–which constitute the core output of this period of study. These works emerged from an extended awareness of the digital domain and the practices linked to this domain–e.g. programming, data and signal processing, and software design. However, this understanding of the digital is not a full explanation for the artistic output since this knowledge lacks the impetus of application. As such, it is from the combination of computer science with the study of visual art–as the motivating force–that these works were produced. Through the study of both disciplines, the awareness of the digital domain was applied to the task of artistic output, focusing on aesthetics and design of artistic works while utilizing the tools, material, and form available through the digital. These works were, therefore, made possible solely through the knowledge acquired in this cross-domain practice.

The creation of these works required an appreciable knowledge of computer science derived from study beyond the realm of visual art. Incidentally, throughout the course of this study, I found many situations when working with visual arts students in the department where the existing tools and platforms were beyond their technical capabilities. While there is obvious value in an intimate knowledge of computation for the creation of digital art works, this level of knowledge is not realistic for the majority of artists and designers when the field of art and the skills required are already so vast. While some tools do exist, as will be discussed in Chapter 2, most of these tools were found to be ineffective within the context of interactive art since

they were either too complicated to be approached without a large amount of previous knowledge or were too limited in their design to be useful for a wide range of artistic outputs. Specifically, I found these limitations detrimental to artistic expression. As a result, I sought to generate a new tool to assist in the creation for my own future works and for use by any artist looking to creating interactive works.

The goal of this masters project is twofold: first, to create a tool for use by artists for the creation of digital interactive installations. Made possible through an open design and with minimal constraints for maximizing flexibility. And, second, to create an interactive installation which simultaneously shows the abilities of this new tool and to act as an art piece in its own right.

This thesis covers two core contributions brought together through the study of computer science and visual art. The first component is a software and hardware platform, *dreamIO*, which was designed as a tool for use by interactive designers and visual artists to solve some of the core issues underlying interactive installation art. The second component is the discussion of an interactive installation, *Transcode*, that I designed and implemented, which acts both as an example implementation for use of *dreamIO* and as a standalone piece.

In the context of digital interactive art, the core problems which *dreamIO* seeks to tackle are scalability, portability, flexibility, performance, and reliability. These issues will be discussed in full in chapter 3. The motivation for tackling these problems comes from the limitations I have experienced while making interactive art–both my own work and in assisting others–and the ineffectiveness of existing tools for handling these problems.

Transcode, discussed in chapter 4, explores one possible implementation utilizing the *dreamIO* framework, through the creation of a piece which is enabled by the features of the system. In this discussion I argue that this installation could not have been created without the framework. Furthermore, along with a discussion of the relationship between the art and the underlying technology, the installation is analyzed and discussed in the context of two viewpoints. These modes of thought, Cybernetics and Relational Aesthetics, are combined together to help understand the relationship between art and technology in the piece and, moreover, as a tool for analyzing and discussing the realm of digital interactive art.

The problems associated with creating interactive art can be very technical in nature and stand in the way of the creation of new and meaningful pieces by acting as a barrier for those without the resources necessary to tackle these technical issues.

While there are tools that can assist some of these issues–discussed in Chapter 2–I have found no comparable tool that is able to tackle all of the problems brought forth in this thesis. In the design of *Transcode*, I seek to navigate these issues through the construction a piece which tackles them head on using the new tool I designed.

The *dreamIO* system mediates these problems though an open-source hardware and software platform. The hardware provides core components for handling interactivity–using sensors and a visual display–as well as wireless communication. The software facilitates the creation of complex interactive 'applications' through handling the core communication mechanisms necessary for creating large networks of devices and by providing pre-built functionality for the artist. This functionality gives artists access to the core resources of the hardware–e.g. sensor data and LED control–with very little effort. This pre-built functionality can be linked together, using practically any programming language, to create larger structures of interactivity.

Transcode uses this wireless system through a complex set of programmed actions, amounting to an interactive algorithm which facilitates a relationship between a real-time visual projection and users through physical manipulation of several *dreamIO* hardware nodes. The design of the piece relies on a vocabulary of actions which are borrowed from music technology. These actions are linked into an interface to create a constantly evolving visual representation of sonic elements directly controlled by the user. At its core, the goal of this work is to provide a constant experience of exploration to users by providing a large set of parameter controls to leveraging the curiosity of users. Thus, the interactivity of the piece is an attempt to facilitate an experience of play and pure experience disconnected from representation.

Through the course of this thesis, several valuable contributions to each domain will be presented. In the context of the *dreamIO* platform, the expressiveness and potential of the system will be shown through presentation of a new approach which directly handles the issues mentioned above. Furthermore, quantitative measurements showing the potential and limitations of the technology will be presented. The system is then be validated through the presentation of *Transcode* as a fully fledged interactive installation, demonstrating the usability of the system through a concrete example. Furthermore, I will show how the combination of two seemingly disparate modes of thought, one from each of domain, can be combined to better understand the piece as well as act as a tool for the creation and understanding of digital art more generally.

1.1 Overview

In Chapter 2, two sets of related work are presented. The first, section 2.1, consists of existing systems for enabling interactivity. Details of these systems are presented along with a comparison with the *dreamIO* system (enabled by data presented in Chapter 3). In the second, section 2.2, two interactive works by Rafael Lozano-Hemmer are discussed. These works are presented to emphasize the complexity consistent with large scale digital interactive installations. The use of *dreamIO* for constructing these installations is also explored as use case examples.

Chapter 3 is a technical document which focuses on *dreamIO*. The design and implementation of the system is presented followed by analysis of the properties of the system. Finally, the feasibility of using the system in real-world application is discussed based on data collected from the system.

In Chapter 4, the design and implementation of *Transcode* is discussed, including specifics involving the physical nature of the piece and the reasoning behind the interaction design. Finally, the piece is discussed in terms of modes of thought, Relational Aesthetics and Cybernetics, merged as a single perspective for understanding the piece and digital art more generally.

Although each chapter of this thesis is equally relevant, Chapter 3 is written primarily from computer science perspective and Chapter 4 more from a visual arts perspective. However, both chapters are informed by each discipline.

Chapter 2

Related Works

2.1 Related Technologies

The *dreamIO* framework is situated in the history of Tangible User Interfaces (TUIs) and Wireless Sensor Networks (WSN). These concepts are explored in depth in Chapter 3.3. In the following section a survey of some of the existing tools and systems are presented. These are tools which attempt to solve some of the same issues as the *dreamIO* framework. Comparison of each system utilizes data presented in section 3.5.

2.1.1 Wireless Inertial Sensor Package

The IMU functionality and wireless data acquisition of the *dreamIO* development board is similar to the functionality available in the Wireless Inertial Sensor Package (WISP) [47]. This device consists of a microcontroller, high quality IMU, and a 900mhz radio for wireless communication. Data collected by the WISP is transmitted to a base station connected to a computer via a USB link. Once the data is received by the base station, it is made available to an intermediary application running on the connected computer from which point it can be sent over a network as Open Sound Control (OSC) messages to any desired application. The device is small, roughly the size of a wristwatch, and is designed to be attached to different parts of the human body for analysis of movement and for use as a real-time control interface.

This system has several major advantages over the *dreamIO* platform: it is smaller, with a volume of less than 23cm^3, compared to the volume of 27cm^3. The mass of the WISP is also less: 23g compared with 33g (see section A.1). The WISP is also capable

of operating for 17 hours on a single charge compared to an average of maximum of 8 hours (see table 3.1). However, there are several disadvantages of the WISP system: the base station and intermediary control software is required to use the system which is a limiting factor compared to the direct OSC messaging capability of *dreamIO*. For data flow, up to four WISPs can can transmit data on a single channel at a frequency of 80Hz, doubling the number of WISPs halves the transmission rate. This is in contrast to *dreamIO* which maintains a constant transmission rate, controlled via the API, effectively invariant from the number of nodes on the system. This invariance is assuming the wireless network is not overloaded. Finally, the WISP system produces only raw data from the IMU and does not handle any pre-processing, effectively requiring the end user to implement all other functionality while *dreamIO* is able to serve a significant amount of pre-processed data (see A.3). As such, the functionality of the WISP is an ideal system for low-invasive measurement of body movement, given its small size and weight, but lacks many of the key features which would help make it accessible for creating interactive works.

2.1.2 AudioCube

AudioCube [43] is a cube shaped TUI for generating and processing audio signals. Each cube contains infrared sensors and emitters allowing the cubes to communicate with one another other, passing audio, via infrared lights, for creating signal processing networks. Users can change the position and order of the cubes relative to one another to control how the cubes' signals flow. Each cube also contains red, green, and blue LEDs to provide optical feedback to users (e.g. what type of signal processing). The cubes contain rechargeable battery packs and their signal processing attributes are configured through a cable connected to a computer application.

This system is a significant departure from the *dreamIO* platform for several reasons: first, the types of interactivity possible with the AudioCube system is limited to the audio domain. In contrast, while the *dreamIO* system cannot inherently generate or process audio, it can control these processes running on an external computer. It is capable of audio and more while AudioCubes cannot be pushed beyond audio processing. The second significant difference is the necessity to pre-program the AudioCubes through a tethering process with a computer, this points to a static relationship between each cube and their functionality. *DreamIO* in contrast, while having static code, has a multitude of capabilities made possible through wireless communication

with the computer(s) responsible for the functionality of the system. In other words, while the code on the *dreamIO* board is static, this code is designed to create larger operations through basic capabilities. The major advantage of the AudioCube system is the ability reconfigure the system based on the cubes' orientation with one another. The ability to re-configure the system via the physical orientation of the cubes with one other is a feature which is not possible with *dreamIO*–which has no way for the hardware nodes to know their physical position in relation to one another. Despite this, *dreamIO* has one obvious advantage: the physical housing for the circuit is not confined to a specific shape. This is in contrast with the predefined form of the AudioCubes.

2.1.3 Arduino 101

Arduino 101 [5] is an extension of the Arduino platform with an integrated 6 axis IMU, 32 bit 32mhz processor, and Bluetooth for communication. This platform comes preloaded with a Real-Time Operating System (RTOS) and is designed for data logging, real-time transmission of sensor data, and can be programmed with the same tools as earlier Arduino models. External sensors and actuators can be added since the platform allows users to write code to handle these hardware peripherals. There are smartphone and tablet applications, as well as desktop applications, which allow for the devices to be communicated with via Bluetooth.

While this system contains some of the same functionality as *dreamIO*, it has a lower clock speed, no onboard RGB LEDs, and requires users to program the device from scratch. Furthermore, while Bluetooth connectivity can significantly simplify device-to-device communication with the pre-built applications, extending the data exchange capabilities is hindered by this communication method. This is because there is significant overhead when designing software which supports Bluetooth and it is limited to devices with Bluetooth hardware. Furthermore, in standard configuration, Bluetooth can only support up to 7 simultaneous connections [9]. This is a significant reduction of the number theoretically possible on a WiFi based system.

2.1.4 Firmata

Firmata [18] is a firmware providing a communication protocol for Arduino (and other microcontrollers) which allows data to be streamed between the firmware loaded microcontroller and a computer. Users do not write code to go onto the microcontroller

but instead have the ability to control the general purpose in/out (GPIO) pins–to get sensor readings, control LEDs, drive motors–from software on the host computer. This software allows for the creation of complex applications through writing software on a host machine, interfacing with the 'real world' through sensors and actuators attached to the microcontroller.

This firmware is similar to *dreamIO* with two major differences: first, editing Firmata to extend the functionality of the system is no trivial task. This leads to software stability and efficiency but is less flexible than *dreamIO*. Second, Firmata only supports USB communication, making the addition of many nodes to a system difficult and the placement of nodes tied to finite cable lengths and available ports on the host computer.

2.1.5 Control

Control [42] is an application which runs on smartphones and tablets. The software allows users to access sensors embedded in the hardware of the device and to stream this data to remote applications. This includes support for front and back cameras, microphone, and IMU. Furthermore, Control handles pre-processing of data which reduces the workload of application development. This includes: onboard musical feature analysis, speech recognition, and video processing (image tracking). Developers can also extend the application, using JavaScript, to extend the functionality of system allowing it to be adapted for broader sets of tasks.

This software has a high number of similarities to *dreamIO* but, given the hardware capabilities of smartphones and tablets, has significantly more features. The flip side of the smartphone reliance is the large overhead in terms of the price of these devices–generally in the hundreds of dollars range for a single unit while *dreamIO* nodes are an estimated $30 CAD. Furthermore, these devices come in a physical form which are not easily augmented. While it is possible to put the components in a custom case the risk of damaging the device is high. In this way, Control can be considered less malleable in terms of physical form. In other words, the software is bound to particular physical parameters and interface which may hinder the abilities of interactivity designers. Conversely, *dreamIO* is designed to be put into custom housing, reducing the physical constraints of design. Without this freedom in terms of form, these devices are likely to be problematic since users will have trouble dissociating the interactivity from their previous experiences with smartphones. This may prevent the creation of new

experiences of interactivity inhibited by users' preconceptions.

2.1.6 TouchOSC

TouchOSC [48] turns smartphones and tablets into customizable touchscreen control interfaces. Users can create different interface templates and set OSC endpoints to map to existing software. The design of this software focuses on creating wireless control for these external programs. The application also allows users to transmit the phones IMU data. While the interface screens are customizable, this software does not allow for preprocessing of input data or extensions of any of the functionality.

This software is similar to Control (discussed above) but lacks the more extensive features while suffering from the same pitfalls associated with being a smartphone application (predefined physical form and price point). Furthermore, this application is paid software and is not open-source, which is in significant contrast with *dreamIO*. In essence, TouchOSC is a limited interface for smartphones and tablets lacking any extended features.

2.1.7 Jam2jam

Jam2jam [15] is a desktop software suit for enabling collaborative audio and visual creation and performance both locally, and over a network, for non-skilled practitioners. The software operates on a standard computer and as such does not require special equipment. The software is designed specifically to facilitate creative collaboration with audio and visuals with an attempt to allow for complex behaviour with very little experience.

This software shares many of the goals of *dreamIO*, including complex collaboration through visual interfaces, but is limited in its functionality beyond its design parameters since the software is not designed to be extended. As such. this software is more likely a candidate for interfacing with *dreamIO*, extending the user interface elements with a wireless controller, than a comparable system. In other words, the spirit of this system is similar to *dreamIO* but with a significantly different purpose.

These related works all live within the same context as the *dreamIO* framework. These system, however, all fall short of solving the goals outlined in this thesis. In other words, I have been unable to find an existing system which can help artists to create complex interactive art applications without requiring a significant amount of augmentation of existing tools.

2.2 Interactive Works

In order to provide context for the *dreamIO* framework, two installations by Rafael Lozano-Hemmer will be discussed. The technical elements of the works will be analyzed and a case for using *dreamIO* for the design of these installations will be presented. The purpose of this section is to reflect on works by a prominent artist in order to explore the complexity which can arise from larger interactive installation and the technical overhead which follows.

2.2.1 Tape Recorders

Tape Recorders (2011) is an interactive installation which consists of a tracking system and a set of motorized tape measures attached to a wall. In the presence of a user the closest tape measure begins to unwind. Each activated tape measure continues unwinding, projecting the tape upwards, until reaching the 3-meter mark. Each hour, the system prints the total number of minutes spent by all users in the space.

There are several components which make this system quite complex. The movement of each tape measure is enabled by individuals motors and some sort of microcontroller to control the movement of these motors. In theory, the sensors responsible for activating each tape measure could be integrated with the microcontroller. This would allow for each tape measure to operate independently (non-networked). If this were the the case, the piece could be constructed out of multiple copies of this hardware and software without any need for communication between elements. However, because the total time of users is tracked, it can be inferred that the system contains some mechanism for communication allowing the total time to be calculated from the tape measure activation.

While I cannot find any metrics on the total technical complexity of this piece, my experience tells me the resources–time and capital–necessary to create this piece are significant. Furthermore, the programming and hardware design of the piece is credited to Stephan Schulz and the expense of hiring a developer is high. Therefore, to create a work like this from scratch would take considerable resources and technical knowledge. Without this knowledge and capital creating a piece of this scale would be difficult.

dreamIO nodes could be easily extended to construct this piece. By attaching a motion sensors and a motor controller each node could act as one of the tape measure drivers. Because of the power requirements of the motor, the nodes would have to

be powered from a power supply but this would be the only cable running to each unit. In the original work, all the cables are hidden behind the gallery wall so cables are not necessarily a concern. The sensor data and activation time could then be streamed to a central computer for generating the hourly user report.

2.2.2 Pulse Room

Pulse Room (2006) is an interactive light work consisting of one to three hundred 300W incandescent bulbs activated by a heart rate sensor. When a user grips the sensor, their heart rate is used to drive the patterns of the lighting. The lights are evenly distributed around the space.

In order to construct this piece, the AC power for the bulbs must be controlled with a digitally controllable switch (most likely a relay). Each of these bulbs would therefore require a cable running to the main control computer. Controlling hundreds of these switches is fairly trivial but the logistics of running the power for this many high wattage bulbs is significant. There are a few conceivable configurations for handling this control. Each bulb may be co-located with their digitally controlled switch and the control signal for each switch would connect to a central control computer. This is unlikely since controlling these switches over a long distance requires special preparation. Another possible option is that somewhere in the space all of the bulbs are connected to a junction box containing an array of these computer controlled switches. This junction box would then be located close to the control computer thereby removing problems associated with long signal lines. In this setup, the length of power cable required for all the bulbs would be significant and therefore costly.

If this piece were created with the *dreamIO* framework, the production overhead, both in terms of cost and time, could be significantly reduced. First, if done without the incandescent lights, the onboard LEDs of the hardware nodes could be used. As a result, each light in the space could be replaced with a single node. The batteries for the hardware could be bypassed with power supplies to reduce the need for charging. However, if incandescent bulbs were used, the nodes could be easily modified with relays to allow them to control the bulbs. As a result, there would be little concern about signal degradation over distance and no need for a centralized junction box. Furthermore, a node could also be modified to handle the heartbeat signal. In essence, using *dreamIO* would allow this installation to be decentralized with mini-

mal technical overhead. The added benefit would be the ability to easily expand the maximum number of lights for the system.

Chapter 3

DreamIO

Wireless Sensor Networks for Enabling Interactive and Digital Art

3.1 Introduction

Digital interactive installations are art pieces which utilize computer hardware and software to facilitate interactivity with users. The form of these installations, including the specific interactivity, can vary greatly in terms of aesthetics, size, and user capacity–i.e. single user or multi-user engagement. Specific examples of two interactive works are outlined in section 2.2.

Facilitating this interactivity requires some combination of:

- sensors, to acquire information from the world;
- actuators, to manipulate the world, and;
- displays, to transmit information to users.

There are a large variety of these devices which can be integrated into interactive software to allow for the input and output necessary for interactivity. Some examples of sensors include motion sensors, temperature sensors, and microphones. Actuators can include electromechanical devices such solenoids or motors. Displays can be a variety of devices which transmit information visually, including computer monitors and LEDs.

Interactive works often require large numbers of these device to be coordinated and controlled over a distance. Even if these distances are small this coordination

and communication is a difficult task because it carries with it a large amount of technical overhead. This overhead includes technical complexity resultant from designing and programming low-level hardware as well as for enabling communication between hardware components. Issues of scalability also come from this complexity since programming for a small number of devices can be easier than programming for a large number. In other words, the complexity of designing software for large numbers of distributed devices is significant and there are few tools designed to mediate this technical difficulty.

This overhead is significant for any artist working with interactivity because as the complexity of an interactive system increases the technical difficulty of implementation also increases. Simply put, as the size of a work is increased–as more elements are added–the difficulties of programming and physical implementation also increase. The technical challenges of scaling stand as a barrier for artists at any level of technical proficiency and impedes the creation of new works. Even for individuals with a high technical skill level the complexity of this task can lead to a large cost in terms of development time and equipment expenses. If an artist's technical proficiency is not sufficient, a possible solution is to hire a technician but this is not feasible for all artists. While technologies for communication and coordination of hardware do exist, these systems–both hardware and software–are designed for other uses and therefore require augmentation–both hardware and software–in order to be utilized effectively within the realm of interactive art. This augmentation is not ideal as it requires significant time and skill. Furthermore, this strategy is not guaranteed to work. To the best of my knowledge, there is currently no other research which focuses explicitly on this problem.

The approach to this problem, outlined in this chapter, is two fold: first, two existing concepts–Tangible User Interfaces (TUI) and Wireless Sensor Networks (WSN)–were combined to create a WiFi enabled hardware platform to be used as a core building block for creating interactive art. TUIs are device which allow for manipulation of digital information through a physical interface [23]. WSNs are networks consisting of sensors–hardware nodes–which acquire data and make it accessible through a wireless interface [3]. Combining these concepts allows for a variety of sensors to be attached with communication between devices enabled through standard WiFi technology. Second, specialized software (firmware) was written to run on the hardware to significantly reduce the development steps for interactive installations by allowing artists to chain functionality–prewritten in the firmware–between these hardware

devices. This software also enables communication with other existing software and tools. This chaining of functionality is facilitated through an Application Programming Interface (API). The utilization of an API for reducing development overhead comes from a re-interpretation of interactive art from the perspective of software development. Through re-imagining interactive art installation as a software design problem, the mechanisms which exist for creating complex software applications become useful in the domain of interactive art. The low-level technical proficiency generally required for hardware based systems is reduced through enabling the use of a large set of high-level software tools. As a result the cost of development for interactive works–technical difficulty, time spent, and capital expense–is reduced.

3.1.1 Claims

Several claims are made which are validated in this chapter:

The hardware and software platform, called *dreamIO*, merges TUI and WSN technologies, coupled with an API, in order to create a system which is:

1. scalable, allowing for large numbers of devices to coordinate towards a common goal;
2. portable, given the small size of the system;
3. has sufficient performance metrics for real-world application, including battery life and data transmission integrity;
4. expressive, made possible through compatibility with a variety of development and software tools, and;
5. flexible, allowing for easy augmentation of the system to extend beyond initial design parameters.

Claims 1, 2, and 3 will be examined quantitatively, utilizing data recorded from the system and claims 4 and 5 will be demonstrated though argument. These claims are important because they point to the validity of this project for use in real-world scenarios.

The results will show that, while the system is not without issues, it has sufficient performance metrics, scalability, and flexibility which constitute a viable tool

for digital interactive art. Furthermore, because of an open and accessible design philosophy, the system can be used beyond the context of artistic practice with potential in wireless data acquisition and real-time data analysis scenarios.

3.1.2 Overview

In section 3.2, various issues related to the design and implementation of digital interactive art installations are discussed, motivating this work. A framework, *dreamIO*, for handling these issues is presented in section 3.3. Implementation details of the framework are discussed in section 3.4. In section 3.5, the methodology for quantitative evaluation of the system is presented, the resulting data from the experiments is brought forth and the data is evaluated in the context of real-world application potential. Finally, future work and extensions to the system are discussed in 3.6.

3.2 Problems in Making Networked Art

The core of digital interactive art, and interactions design more generally, is sensors, actuators, and displays–user input and output; control and feedback. These inputs and outputs are made possible through physical hardware coupled with software systems for processing and controlling data. This digital interactivity is now common, taking place through web applications and enabled by the ubiquity of personal electronic devices such as smartsphones and laptops. These devices have been leveraged for enabling interactivity in recent years [42, 28, 36]. Using these existing tools allows interactivity designers to leverage the large number of sensors available on these devices (e.g. image sensor, inertial measurement unit, global positioning system, etc), the significant processing power, wireless communication, and Internet enabled technologies (e.g. javascript, multi-device communication, video streaming, etc).

The core components necessary to create web based interactivity are readily available and in some cases leveraging these existing technologies for the creation of interactive installations is possible. However, there are limitations for these adaptation practices. For example, the high price point of these devices can make their use in a gallery setting impossible. To counter this issue, one may utilize the fact that smartphones are ubiquitous and require participants to bring their own devices. However, this solution is precarious as the diversity of devices available on the market can

make development of interactive applications difficult. Different makes and models of devices can have radically different sensors and adding new and custom sensors to mobile devices is a significantly difficult task. Along with the physical differences in hardware, there are a wide range of operating systems running on these devices and designing applications which are stable across many different platforms is a major undertaking. Extensive software testing for multiple devices is time consuming and is a fairly advanced software development technique. Even getting access to every potential device for the purposes of testings is unlikely. While cross platform solutions do exist–e.g. JavaScript inside a web browser–even these can be impeded by the diversity of devices. Therefore, despite their presence in modern society, smartphones and tablets are not always a good choice for developing interactive art installations.

Another potential solution is the use of common user interface devices such as keyboards, mice, and video game controllers. Dependent on the capabilities of the device they can be easily connected to a computer through a USB connection or wirelessly via Bluetooth. The functionality of these devices can then be mapped to interactive elements using Max/MSP[1] or other programming languages. However, without physical augmentation, the form factor of these devices can be a determent. They can carry predefined behaviour linked to their use in everyday life. In other words, because these devices are common in the modern world, an artist may find it difficult to generate new experience or interactive modes. Furthermore, the limitations of wired connections (discussed below) and of Bluetooth (discussed in Chapter 2) can be detrimental to the design and implementation of a piece.

Cameras can also act as an excellent control interface for interactive works. However, there is a large amount of technical overhead when designing computer vision software to track motion. The Microsoft Kinect [27] system reduced some of the overhead with camera control by providing automatic skeletal tracking. There are limitations to this tracking, including issues with multiple users and occlusion (when a user is obscured from view). Furthermore, it is difficult to use multiple Kinects simultaneously without significant technical overhead.

A common alternative to these methods is the use of microcontrollers. These devices are small, low powered, and inexpensive computers which allow for the programming of input and output necessary for interactivity. One such example of an accessible microcontroller is the Arduino [4] which has become a favourite of hobby-

[1]Max/MSP is a high-level multimedia programming language designed for real-time and interactive audio and visual expression [32].

ists and artists. Arduino is an open-source microcontroller board and an integrated development environment (IDE) for programming functionality for the the board. It was designed as a low-cost and simple tool for creating digital projects [6]. While this tool, and other like it, have significantly reduced the overhead for interactive works, there are still several issues which can impede the development of interactivity.

Using large numbers of sensors and actuators is a difficult task. Given their low computational power and physical constraints in design, for significantly complex tasks–e.g. audio signal processing–or large numbers of inputs and outputs, microcontrollers can very quickly reach their limitations. It is possible to use many microcontrollers by connecting multiple devices together but this also brings significant challenges both in terms of software and hardware development. Communication between devices has to be enabled through some physical interface–e.g. radio communication, Ethernet port, infrared–which many devices simply do not have. Furthermore, writing software for aggregating multiple data streams and creating complex interactivity between devices is a significant task. Code must be written for each device in the system and in order to make sure the code runs smoothly it must be tested both individually and as a component in the larger system. The totality of this work amounts to a significant task leading to large costs. These costs can be in terms of time spent developing, even for those with a high level of technical knowledge, or financial costs associated with hiring technicians.

Some of theses issues can be solved through linking microcontroller systems into more capable computer systems: i.e. laptop or desktop computers. Many microcontrollers are equipped with USB serial interfaces which allow the devices to connect to a host computer for sending and receive data. This setup alleviates some of the issues with programming complex interaction between multiple devices by allowing for the interactivity to be programmed on the host computer with much more powerful tools and programming languages–e.g.Max/MSP, C++, Python. In this setup, the microcontroller acts as an interface between users and the host computer. While this does reduce much of the overhead associated with creating interactive works it also poses significant challenges: A communication protocol–a standardized way of encoding and communicating information between devices–must be used and all the devices involved must be programmed in such a way as to handle this protocol. While there are existing tools that handle this protocol challenge–discussed in Chapter 2–these solutions are not always optimal. This points again to the difficulties of writing code for a complex interactive system made of distributed elements.

Communication protocols aside, USB communication also has physical limitations: First, the microcontrollers become tethered through the cables. This is not a significant problem for all works but can impede any interactivity which requires users to physically move the sensors attached to the microcontroller. Second, there are finite USB ports to connect to on a computer and so for larger installations USB hubs have to be added in order to increase the capacity of the system. Third, having a large number of cables running around a space can be prohibitive and an aesthetic challenge in terms of visual clutter. Finally, the maximum length of a USB 2.0 cable is limited to 5 meters [14]. For larger installations, these negative attributes can be a significant detriment and easily outstrip the reliability of USB communication. USB tethered user interface devices (keyboard, mice, etc) also suffer from these issues.

Another significant issue with the existing solutions for interactive art systems is the physical and aesthetic properties of these devices. Smartphones, tablets, and common user interface devices have very specific form factors and are a source of pre-conditioned interaction for users. In other words, the shape and meaning of an everyday electronic device can impede creative expression by reducing the affordances available to users because of their everyday use and context. This has the potential of significantly reducing an artists expressive potential when designing their interactivity and can limit visual aesthetics of a piece. Taking a smartphone's components out of the case and placing them in a new form is certainly possible but cumbersome.

Microcontrollers are significantly more pliable in terms of form but most do not allow for the same level of complexity made possible through the integrated sensors and actuators available in tablets and smartphones. And, unlike smartphones and tablets, few microcontrollers come stock with the ability to communicate wirelessly and, for the small number that do, there are few software tools in existence to facilitate complex interaction of many units in real-time. Once again, both solutions fall short for use in this specific context.

To reiterate, existing interfaces for interactive art have several major issues: the overhead for development can be prohibitive, using existing interfaces can limit interactive and aesthetic potential, and networking large numbers of devices is difficult. While microcontrollers can help facilitate these complex interactions, they are limited in their ability to communicate in an easy and effective manner. Furthermore, aside from the utilization of smartphone and tablet technologies, which are rigid and expensive, wireless communication tools for the purposes of interactive art are essentially non-existent. Currently, to the best of my knowledge, there is no solution which

satisfies all of these requirements.

3.3 Designing for Interactivity

In order to solve the above problems a hardware and software 'platform', called *dreamIO*, was created. This platform utilizes off the shelf electronic components (microcontroller, sensor, power management, and LEDs), open-source software libraries, and a standardized wireless communication protocol. These components are brought together through a combination of three existing mechanism in an effort to solve the issues discussed in the previous section. The three mechanisms are presented below:

3.3.1 Tangible User Interface

A Tangible User Interface (TUI) is an electronic device allowing for the manipulation of a digital environment through a physical interface. TUIs allow users to interact with digital systems through manipulation of physical objects [23]. A rudimentary example of a TUI is a computer mouse and keyboard, which allow users to control and manipulate the virtual environment of their desktop. Through specially created hardware and software, TUIs allow users to directly manipulate a virtual space through physical action. They are valuable for generating interactivity through digital-physical interface. TUI design, however, does not implicitly define any mechanism for multi-device communication, therefore in order to create larger and more complex works using TUIs, some mechanism must be utilized to allow these devices to communicate together, to share their information in a useful way.

3.3.2 Wireless Sensor Networks

Wireless sensor networks (WSN) are made up of low-power, multi-functional electronic sensor nodes which communicate data through a wireless interface. Advanced WSNs can have nodes with multiple sensors and complex processing capabilities. Depending on the complexity of the network, WSNs can be comprised of a few nodes to thousands [3].

In the past, WSN nodes have been limited by the available technologies. Recent advancements, however, in small-scale digital devices have made the development of WSN platforms more feasible and significantly less complicated than would previously

have been possible. These advancements are rooted both in physical hardware and underlying software.

The scalability and flexibility of WSNs is ideal for leveraging a large number of sensors and displays, and for facilitating communication. On the other hand, having nodes communicate in a network does not presuppose application. By combining the physical nature of TUIs with the wireless communication and scaling capabilities of WSNs, a diverse range of potential combinations is made possible. However, the product of these two concepts is limited without some mechanism for the design and configuration of complex and dynamic interactive systems. This can be accomplished through defining an interface for communication.

3.3.3 Application Programming Interface

An application programming interface (API) is a mechanism which allows for the creation of complex software through assembling pre-built functional units of code into larger structures. This is enabled by providing a communication mechanism–an interface–for passing data between these units of code. In essence, APIs specify how different software and hardware components should interact through a defined language. The API construct is a powerful mechanism for creating complex applications because much of the code is already written and so an application can be assembled through gluing this code together. In essence, an API acts as a bridge between modules of code, allowing for the design and construction of higher level functionality–i.e. applications. In the case of WSNs, a shared language can allow for interactivity designers to acquire data from sensor nodes, process this data, and route this processed information to any node on same network. Because the nodes on the network all speak the same language, application designers are not required to reprogram the individual nodes in the system but instead can create programs which utilize these wireless units. This mechanism also reduces the amount of code written since it allows identical code to be placed on all nodes. This is possible since identical functionality can be present on all node but activated only when necessary. In other words, because the behaviour of the system is defined externally–through linking of existing units–the same code can be activated in different way. By utilizing the API mechanism, the total complexity of designing applications is significantly reduced since designers need only to worry about connecting the functional units together and do not need to consider how the underlying architecture of the system is

constructed.

3.3.4 Merging Concepts

While both TUI and WSN systems have been previously utilized for collaborative interactive art [19, 20, 35, 47], to the best of my knowledge there is currently no system which combines the two concepts with a comprehensive API. The *dreamIO* platform is an attempt to merge these three concepts to create a software and hardware framework designed to reduce the overhead for the creation of interactive art applications.

The platform is implemented as a hardware and software package. The hardware consists of an inexpensive (less than $5 CAD per unit) WiFi enabled microcontroller integrated with an inertial measurement unit (IMU) on a single low profile circuit board. The IMU is a sensor which enables the microcontroller to measure acceleration and rotational velocity of the circuit board allowing it to be used as a physical interface, i.e. direct manipulation through physical movement. By combining an IMU and WiFi capabilities, this hardware is able to act simultaneously as a TUI and WSN node. Using a standard WiFi router, many of nodes can connect to a a shared wireless network.

Specially designed software–firmware–was written for the microcontroller in order to automatically handle management of the microcontrollers functions including collection and processing of data from the IMU. The software enables these nodes to communicate over a WiFi network via the API. The API is implemented using a standardized communication protocol–Open Sound Control (OSC) [56]–allowing a large diversity of different hardware and software tools to communicate with nodes on the network. This compatibility allows the system to leverage an ecosystem of existing tools and resources.

Using a common language–the API–and a wireless communication standard–WiFi–development of interactive applications is enabled across a diversity of hardware platforms (PC, Mac, Raspberry Pi) using a large variety of tools and programming languages (C++, Max/MSP, Python, JavaScript). By definition, any computer system capable of connecting and communicating over a WiFi network is capable of communication with any of the nodes on the network–either on a local network or through the Internet. This system allows the application designer–the artist creating the interactive installation–the ability to define how each node operates in a non-static

way. The behaviour of the nodes can be changed easily because their behaviour is defined by the code written by the application designer. This is in significant contrast with a system in which the behaviour of each node would have to be predefined. Any change in systems behaviour would require code re-writes for each node in the system. In other words, the application is enabled through the functionality of the nodes–coded as the firmware on the nodes–but the behaviour of the system is not fixed or static since it is created through linking each nodes' functionality together via the API. Furthermore, this design allows for the nodes in the system to control and to be controlled through compatible devices on the network. For example, an application designer could easily link a node to control the amplitude of audio playing from a laptop computer. In other words, the WiFi based API allows for communication beyond just the nodes of the system.

While the mechanism of application programming via an API is common in software design, there has previously been little or no effort to combine this mechanism with dedicated TUI/WSN hardware. By doing so the intention is to greatly reduce the overhead required for creating complex interactive installations through bypassing the need for embedded software development and hardware design. Skills which are very specialized and do not necessarily contribute to interactivity design. With *dreamIO*, end users can write complex applications, with few (one) or many (hundreds) nodes, in the programming language of their choice running on whatever computer system they want. While the firmware is designed to be open and easily expendable, out of the box no effort is required. Given the complexity of the task, *dreamIO* constitutes a significant reduction in production overhead compared to building an equivalent system from scratch.

Additionally, because of the use of OSC, the platform can be used as a control interface, through physical motion of the device, with a multitude of existing applications which support this protocol–including Resolume Arena [41] and Max For Live [31]. This allows the platform to act as a human interface device 'out of the box', with the ability to directly control parameters within these software tools using physical movement of the interface.

My goal in merging these concepts is to increase the accessibility of these technologies to a greater portion of artists. The physicality of the TUI, as an embodied interface, is ideal since it allows for nuanced control beyond discrete on/off switching actions. In other words, this interface can enable interactivity through embodied action from physical movement and touching which amounts to a haptic experience for

end users. By making the system operate like a WSN, I hope to reduce the development complexity which occurs with scaling to large numbers of sensors. In essence, the system reduces development overhead through creating the basic units necessary to construct an interactive work. This amounts to an increase in accessibility to the tools and reduction of technical skill necessary for creating complex interactive works.

3.4 System Details

3.4.1 The Development Board

The *dreamIO* platform is fundamentally a framework for creating interactive installation. In order to implement this framework a development board has been designed. This development board consists of a WiFi enabled microcontroller, an IMU, individually addressable RGB LEDs, and components necessary for handling the power requirements of the circuit. Specifications for this hardware can be seen in section A.1. This circuit serves as a tool for continuous development of the firmware, as an example implementation of the hardware (for future hardware development), and as a tool for creating interactive works.

While there are several Wifi enabled microcontroller development boards on the market (see A.2), I was unable to find a device which matched all of the criteria necessary for the specific goals of this application. Using an existing microcontroller board would have required the creation of a extra hardware to extend its functionality. The difference in development time between designing a microcontroller board from scratch–with all the necessary features–compared to an extension to existing hardware is negligible. Given this small difference, I opted to design the *dreamIO* development board from scratch. This reduced the overall price of development, decreased the size of the printed circuit board (PCB), and allowed for the design of a specific form factor. The estimated cost of the board is around $30 CAD which is about the same as an Arduino Uno microcontroller. If a hardware extension board was designed it would have cost close to the same price but would also require purchasing a WiFi microcontroller board. The PCB was designed using KiCad [26], an open-source electronics design software.

3.4.2 Hardware Details

Below is a detailed description of the implemented hardware.

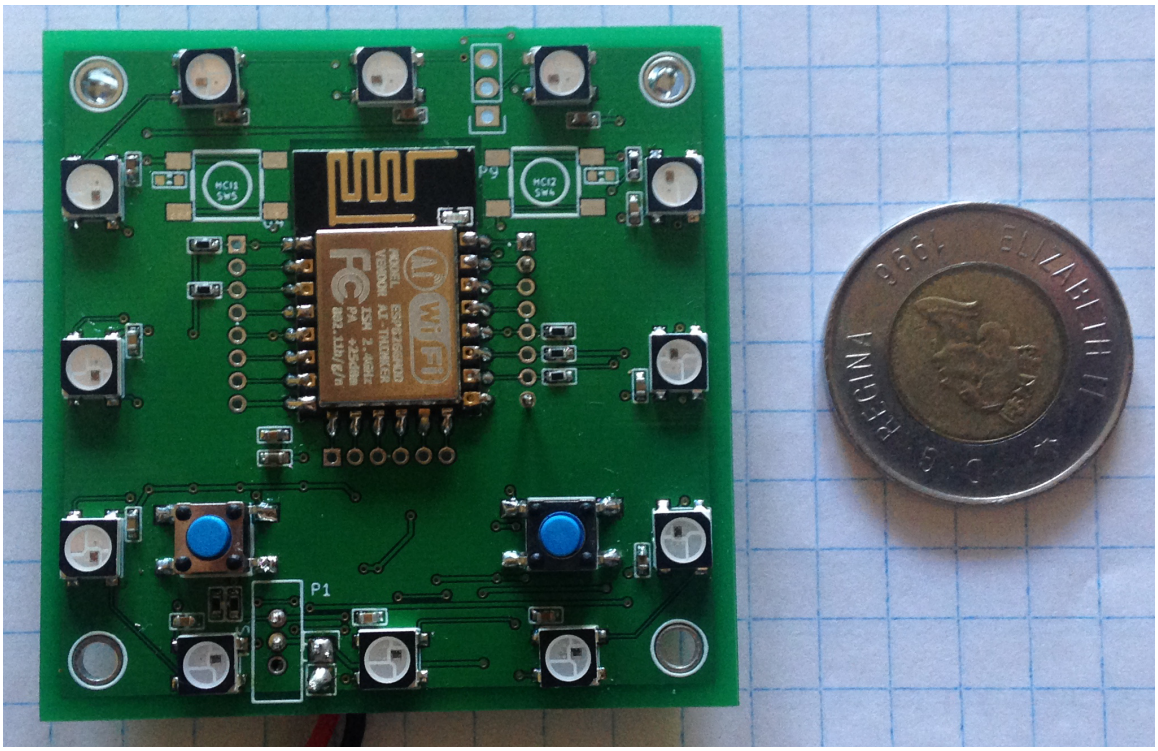

Figure 3.1: DreamIO hardware next to a Canadian two dollar coin for scale.

An ESP8266-12E [17] WiFi enabled microcontroller module–manufactured by Espressif Systems–acts as the brain for the circuit. The microcontroller is a reduced instruction set computer (RISC) based 80Mhz processor with expandable memory ranging from 512kB to 4MB. WiFi is enabled on the microcontroller through a hardware implementation of the TCP/IP stack. This feature greatly reduces the computational overhead required for running WiFi because the majority of the work required for this access is done by dedicated hardware on chip. The module has 16 general purpose in/out (GPIO) pins which can be used to interface with external devices. These external devices include sensors, lights, and mechanical devices such as motors and solenoids.

The main sensor for the board is an MPU-6050 [22] IMU manufactured by InvenSense. This chip contains a MEMS 3-axis gyroscope configurable to 250, 500, 1000, and 2000 degrees per second and a MEMS 3-axis accelerometer configurable to ranges of 2g, 4g, 8g, and 16g. Both the gyroscope and accelerometer have a 16bit

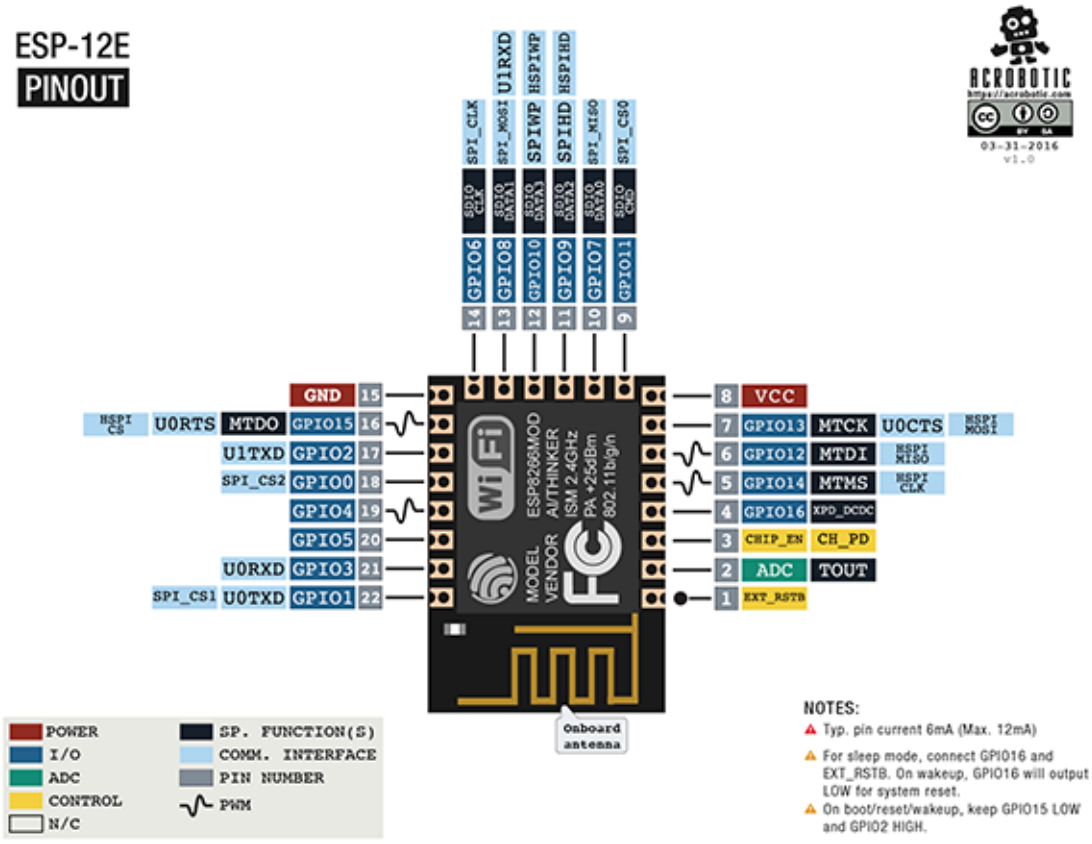

Figure 3.2: ESP8266 12E Module with Pinout. Copyright 2015. ACROBOTIC Industries. All Rights Reserved.

analogue to digital converter (ADC) which provides the digitized sensor data to the microcontroller. This chip enables the circuit to measure relative motion: angular velocity and directional acceleration which can be used to measure physical motion of the device.

The board is equipped with a an onboaerd display consisting of 12 WS2812b [55] LED modules. Each LED module contains a red, green, and blue LED and is individually addressable. This allows for the brightness of each colour channel on each LED to be controlled independently by the microcontroller. Through varying the brightness of each LED colour channel a large number of output colours can be produced.

Because the board is designed to operate wirelessly, two power management circuits are included on the PCB. The first is the MCP73831 [34] charge management controller. This integrated circuit (IC) is a Lithium-Ion/Lithium-Polymer (LiPo) bat-

tery charger, allowing a battery attached to the circuit to be charged via a standard USB micro cable. The chip is configured to draw up to 500mA from the external power source.

The second component of the power management system is a TPS61200 [45] boost converter. This circuit provides two functions: first, this chip regulates the voltage coming from the battery and outputs a constant 3.3V. This is used to power the LEDs, IMU, and microcontroller. The second function of this chip is under-voltage protection for the battery. LiPo batteries can be damaged if they are discharged to below 2.6V. This chip monitors the battery voltage level and shuts down the circuit when it reaches this threshold.

The PCB is also designed to give the ESP8266's onboard ADC the ability to measure the voltage of the battery. The ADC has an input range of 0V to 1V while the battery has an effective operating voltage of 2.6V to 4.2V. To compensate for this difference in voltage the circuit contains a voltage divider in order to scale the voltage before reaching the ADC. Since the voltage regulator outputs a constant 3.3V, the battery is connected directly to the ADC through the voltage divider to get a direct measurement of the battery voltage. The calculations for the voltage divider can be found in the appendix at A.4.

The configuration of circuitry for each chip was implemented based on the datasheet supplied by vendors.

3.4.3 Firmware

The functionality of the *dreamIO* platform is provided by specially written firmware– the core software running on the hardware device. This firmware is written in a combination of the C and C++ languages and is compiled and flashed to the ESP8266 microcontroller using the Arduino IDE. A number of open-source libraries were utilized to reduce the development time of the firmware. These libraries can be found in section A.6.

The firmware is responsible for three core functions:

1. Communication with the low-level hardware, including:
    - WiFi hardware (built into the microcontroller)
    - LEDs
    - IMU

- Analogue to Digital Converter (ADC) for measuring battery voltage

2. Handling data, including:
   - storage of IMU and battery voltage measurement data
   - feature processing (more on this below)
3. API calls, including:
   - formatting data for broadcasting
   - routing incoming messages

While the firmware is intended to be static, the source code has been made available (open-source) and is designed to be easily extended with additional functionality. The firmware is compatible with any ESP8266 based boards but will require modification in the event of discrepancies or changes in hardware attached to the board (sensors and display). All ESP8266 based boards listed in appendix A.2 should be compatible with the firmware however none of these boards have been tested.

3.4.4 Feature Extraction

Features are the measurements of properties for observed phenomena [8]. In the context of this platform, features define information about the physical state of the board. By calculating the features onboard, and making the values accessible via the API, complexity of designing end applications is reduced. The majority of the features calculated are derived from the data measured by the IMU. The features available in the firmware are:

Rotational attributes the relative rotational orientation of the board–defined as *yaw*, *pitch*, and *roll*. This data describes the rotation of the board around the X, Y, and Z-axis and is derived from the velocity and acceleration measured by the IMU.

Orientation relative to gravity This feature indicates which side the board is sitting. The measurement is output as an integer from -1 to 5 mapped to sides as indicated in table A.2. This feature is derived from the acceleration measurement from the IMU: gravity pulls at a constant 1g as measured by the sensor which gives an indication of which side of the board is oriented downwards.

Motion detection This feature indicates if the board is in motion. Motion is derived from the acceleration values from the IMU. After accounting for gravity, if the acceleration in any direction is non-zero–above an empirically chosen threshold–then the board is known to be in motion. Once the board has been at rest for at least 500ms the system indicates a non-motion state.

Battery level This feature is a modification of the recorded battery voltage. As described in appendix A.4, the measured voltage is scaled to a value between 0 and 1, which effectively acts as a percentage indicator of battery level.

The intention is to expand the available features in future versions of the firmware. Expanding the feature set can be bootstrapped with the system itself by prototyping new features through real-time data streamed via the API. In essence, the same processes and tools for creating interactive applications with *dreamIO* can also be used to improve its functionality. Once a feature is prototyped it can then be ported to work directly as part of the firmware. To reiterate, feature development can be assisted with the existing hardware, tested and refined with real-time data, and later implemented into the firmware. This has a significant value in terms of future development of *dreamIO*.

3.4.5 The API

The API uses the OSC protocol for communication to and from the device over a standard WiFi network. All of the functionality of *dreamIO* has been mapped to OSC endpoints which constitutes the API. This allows users to create applications on a computer system (PC, Mac, Raspberry Pi) in basically any language which supports OSC communication. Applications can be composed and interfaced with essentially any other tool or software suite which runs on the user's chosen platform. For example, users wanting to control music parameters in Ableton Live [1] can easily integrate the IMU information from the *dreamIO* development board with a Max for Live [31] patch. This would allow for control of volume, effects, or any other parameter available in the software.

The Open Sound Control protocol utilizes a plain text addressing system similar to the Hypertext Transfer Protocol (HTTP) [7] A forward slash followed by a keyword is used to indicate different endpoints in the system. Specific endpoints are defined in the firmware and are mapped to different sections of code (functions). Thus

OSC messages become mechanisms for calling functions–and passing data to these functions–from applications running on the WiFi network. This is the underlying mechanism which allows for data to be passed to and from nodes in system, enabled through the API. It can be helpful to think of the API as a way for code running on different systems to communicate with each other through the network.

Along with inputs, the *dreamIO* firmware is responsible for outputting sensor data and extracted features. This data is sent as OSC messages with predefined endpoints. This allows applications running on the network to receive sensor data and features from the hardware nodes.

The firmware is configured to allow zero configuration of the device upon boot. This is achieved through a combination of methods:

First, each *dreamIO* board is configured to send UDP packets (OSC messages) to the broadcast IP address[2] on port 9999. This configuration allows every device on the network listening on this port to receive OSC messages from the board. The benefit of this mechanism is that any application connected to the network will be able to receive messages rather than an exclusive device-to-device connection. The drawback of this configuration is that with each device added, the network router must re-transmit every received packet to every other device on the network. While not investigated, this is likely to result in degradation of communication performance. This is a problem which will be resolved in the future through the creation of an API endpoint for reassigning the output IP address for each node. This will allow for the option of creating exclusive connections between the application and nodes in the system.

The second method which allows for zero configuration is mDNS [33]. This service enables each board on the network to be discovered through the mapping of a 'hostname' to the devices IP address. Each ESP8266 has a unique identifier and this is used as the hostname for creating a connection between an application and a node, specifically:

$$esp8266 - [deviceID].local$$

This mechanism is necessary since, without configuration of the network router, IP addresses are assigned when a device connects to the network and are not guaranteed to be identical on each boot. Any application wishing to send OSC messages to a

[2] 255.255.255.255

board can use the device's hostname instead of an IP address. This allows applications to send messages directly to individual boards and allows the connections to persist even after a power reset.

With these two methods, applications are able to begin sending and receiving data from each board immediately. While configuring of the system through a software tool would not be a detriment, the major benefit of this zero configuration scheme is that any existing application with support for OSC messaging will be able to communicate with nodes in a system by listening to the specific endpoints of the API. This makes *dreamIO* nodes capable of acting as physical control interfaces for a multitude of software with very little configuration.

Because the firmware is setup for all boards to send UDP packets to port 9999, a mechanism is necessary to differentiate the messages from different board on the network. Each board's unique ID is used at the base of the OSC address for each message sent from the device.

For example:

$$/[deviceID]/[endpoint]\ [data]$$

This address configuration allows applications to differentiate between received messages from multiple dreamIO devices on the same network.

The API of the current version of the firmware has three OSC endpoints for receiving and four for sending. A complete list and description of these endpoints can be found in the appendix A.3.

3.4.6 WiFi

WiFi is the core technology for enabling the development of such a platform. While other wireless options are available, the use of this standardized communication method has a lot of advantages. WiFi is supported in a huge number of devices, which allows the *dreamIO* platform to communicate with, control, and be controlled by other systems. With alternative wireless systems, a base station would have to be designed to bridge communication to other platforms. Furthermore, by building a layer–the API–on top of the existing protocol, a standardized and cross platform way of communicating is achieved. The flow of data in the system can be seen in Figure 3.3. Furthermore, WiFi is commonly available across many platforms–allowing interaction with the API from a large number of different devices–and can support a large number of sensor nodes made possible through high data transmission rates.

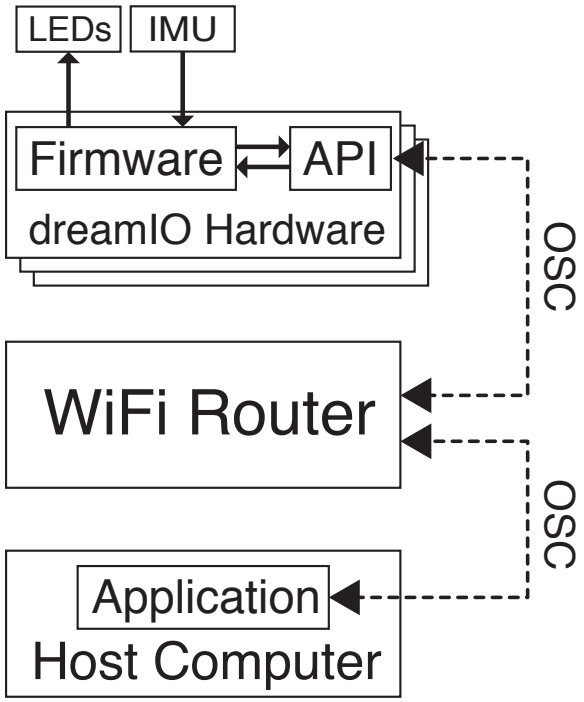

Figure 3.3: *dreamIO* system graph. Note: the layering of *dreamIO* panels represents multiple devices connected to the network.

3.4.7 Future Improvements

The use of OSC has made the addition of new API endpoints a trivial process. Therefore, adaptation of the firmware to incorporate new sensors and features is an easy task. While modification is available (and highly encouraged), zero modification of the firmware is required out of the box.

Several improvements are planned for firmware upgrades. First, two other modes of messaging, an event based mode, in which the dreamIO board only transmits data when an 'event', some change in state predefined by the user, occurs. For example, an event could be a change in yaw values. This mode would allow for the creation of responsive applications, acting on data only when it's made available.

The second messaging mode is a request based model. In this mode, only the data requested by the application, via OSC endpoints, is sent by the board. This should decrease network traffic and may be especially useful for large numbers of nodes.

Furthermore, to increase usability, alternative communications mechanisms will be investigated, such as HTTP over TCP, to be used in tandem with the OSC protocol.

3.4.8 Extended Use Applications

While this tool was designed primarily for facilitating interactive art, there are several attributes of this platform which allow it to be used for other applications. First, because data can be streamed in effectively real-time from the board, and the board is quite small, it can be attached to people or objects to capture movement. These movements could then be mapped to any number of outputs–LEDs on the board, large digital projection, or musical tones, to name a few. This has potential in performing arts such as theatre or dance.

Applications are not limited to real-time interaction. This tool can be used for data acquisition with real-time analysis. This is opposed to a device like a data logger in which data is stored locally on the device and then processed later. For example, if one was to record the motion of a hoola-hoop with an IMU and datalogger, the data would only be available after the recording. Conversely, with this system, the data can be broadcast allowing for analysis with real-time streaming data. Furthermore, this platform can be configured to work through the Internet. This could allow artists and scientists to easily acquire data from sensors in remote locations and over large distance.

3.5 Experiments

The following section outlines the experiments conducted and data acquired for evaluating the system. It is possible to skip this section.

Experiments were conducted to acquire metrics for real-world applications. Furthermore, the process of recording the data, explained below, utilizing a Python script , constitutes a design reference for using *dreamIO* with Python. The Python script is included in appendix A.7.

3.5.1 Methodology

In order to measure characteristics of the development board, a command line script was created using Python [39] to capture data streamed from a single board. The script used for this is available in appendix A.7. Using the same board for every recording, mated with the same battery, acted as a control in the experiment. The Python script was run on a Raspberry Pi 3 [38], running Raspian Jesse Linux [24] and connected to a DLink DIR-625 [16] router. The Raspberry Pi and the tested

board were the only devices on the network. This was done to isolate the test from interference and potential network latency caused by other devices. The Raspberry Pi was configured with a hardwired network connection to reduce the potential for connection dropouts. This configuration represents a realistic real-world use case scenario.

The board was charged fully before every recording, disconnected from the charging circuit moments before the beginning of each experiment. After the board connected to the network the Python script was started manually. This introduced a small amount of time variance in recording lengths but, given the timescale of the recorded data, does not constitute enough variance to contaminate results. The script started by sending API commands to the board to configure onboard parameters for the experiment–broadcast rate and LED colour values were configured at this stage. After the script sent the configuration messages, it immediately began to record incoming UDP packets sent from the board. The UDP packets (OSC Messages) were recorded into a 256 sample sized buffer. The buffer was appended to a comma-separated values (CSV) file–created at the beginning of the script–for later analysis. The use of a buffer helped prevent blocking of the Python script by reducing the frequency of writing data to disk. The script was configured without a timeout and recording of data continued until the board stopped transmitting which signified a dead battery. After the data finished transmitting the script was manually exited which finalized the data recording process through saving any data remaining in an underfilled buffer before closing the CSV file.

To prevent misplacing data, the filename for the CSV file was automatically generated with the script, recording the development board ID, date and time when the script started, the board broadcast interval, and the LED brightness for each colour channel.

The data collected from the development board corresponds to the regularly accessible API outputs with the addition of a special debug value, 'packet count', which incremented after every UDP transmission from the board. If a packet was missed, the recorded value for this count would appear to increment by a value greater than one.

While there may be issue with this method for acquiring data–i.e. there is no guarantee that the data sent from the board is accurate–the exact values of these results are less important than the overall usability of the device. In other words, this method constitutes a use case of the development board and therefore the results are

Table 3.1: Battery Life vs Broadcast Rate

Broadcast Rate (ms)	Avg Time Active (hrs)
10	7.615
25	8.083
100	8.262
1000	8.001
10000	7.988
Total Average	7.990

Table 3.2: Battery Life vs LED Brightness

LEDs	Avg Time Active (hrs)	Min Time Active (hrs)	Max Time Active (hrs)
Red	2.665	2.581	2.891
Green	2.724	2.616	2.957
Blue	2.751	2.618	3.003
Total Average	2.713	2.605	2.950

reflective of what an end user can expect when developing applications with this tool.

3.5.2 Results

Battery Life

The goal of this experiment was to determine the battery life in relation to different configurations. Battery life, in this context, is the length of time the system operates from a fully charged battery.

Table 3.1 contains the average of two data measurements for each of the broadcast rates specified.

Table 3.2 compares the battery life with each LED channel with 100% brightness. The dataset contains the average of five data measurements for each of the LED channels.

Packet Loss

The purpose of this experiment was to measure the packet loss in relation to different parameter settings. In this context, packet loss is the incomplete transmission of a UDP message from the device.

Table 3.3 compares the packet difference, which is the difference in packet number between two samples. A packet difference of 1 means between two sequential data

Table 3.3: Packet Difference vs Broadcast Rate

Broadcast Rate (ms)	Avg Packet Diff	Max Packet Diff	Std
10	1.00025	473.0	0.328
25	1.00002	12.5	0.012
100	1.0	1.0	0.0
1000	1.00005	1.5	0.005
10000	1.0	1.0	0

Table 3.4: Packet Difference Distribution vs Broadcast Rate

Broadcast Rate (ms)	Number Below Avg + 1Std
10	0.999983
25	0.999993
100	1.0
1000	0.999948
10000	1.0

frames no messages were lost. This data is comprised of the average from two data measurements per broadcast rate.

Table 3.4 compares the packet difference distribution, which is defined in this context as the number of packet difference values which are below the average plus one standard deviation. The purpose is to give an indication of the distribution of the data without a full histogram, showing the percentage of packets which are close to the average.

Broadcast Delay

The purpose of this experiment was to see the total broadcast time against the update rate set on the device. This attribute defines the latency of broadcast messages from the device to the host application. The total latency of the system for a round trip message–to the host application and back–is expected to be twice the broadcast delay of the device however this has not been tested directly.

Table 3.5 shows a comparison of the broadcast delay and the broadcast rate to see the difference between the timing settings on the device and the actual time of transmission. In this context, difference is defined as the difference between successive data frames, as calculated from the timer on the device. The measurements are the average of two recordings per broadcast rate.

Table 3.6 shows the number of packets with timing less than both the average plus one standard deviation and two standard deviations. The purpose is to give an

Table 3.5: Broadcast Delay vs Broadcast Rate

Broadcast Rate (ms)	Avg Diff(ms)	Min Diff(ms)	Max Diff(ms)	Std Diff
10	12.8	12.0	6811.5	5.477
25	29.7	27.0	372.0	0.802
100	106.7	29.0	111.5	1.866
1000	1006.8	515.0	1511.0	9.491
10000	9993.7	29.0	10008.5	321.889

Table 3.6: Broadcast Delay Distribution vs Broadcast Rate

Broadcast Rate (ms)	Number Below Avg + 1Std	Number Below Avg + 2Std
10	0.98616	0.99998
25	0.89387	0.99731
100	0.87212	0.99849
1000	0.99995	0.99996
10000	1.0	1.0

indication of the distribution without a full histogram.

Current Draw

The purpose of this experiment was to examine the relationship between the current draw of the board and the brightness of the LEDs.

All current draw data was recorded by supplying 3.7V to the device using an Agilent E3631A [2] laboratory power supply. This device measures current with 0.001A precision. The brightness level for each LED is defined as a software value ranging from 0 (off) and 255 (full brightness).

Tables A.3, A.4, and A.5 show the current draw for each colour channel independently in increments of 10.

Table A.6 shows the current draw for all colour channels with equal intensity in increments of 25.

Data Rate

Wireshark [54] network protocol analyzer–a software tool for recording and analyzing network data–was used to record packets from the device running at an update rate of 25ms.

The purpose of this experiment was to acquire a high resolution of data pertaining to network bandwidth.

From full battery charge, the device lasted 7.71 hours and sent a total of 941,412 packets.

Using Wireshark's built in utilities, several pieces of information were calculated: The length of each UDP packet for the OSC messages is 206 Bytes for all received packets. The data transmission rate for this single device was 41 kilobits per second (kbit/s).

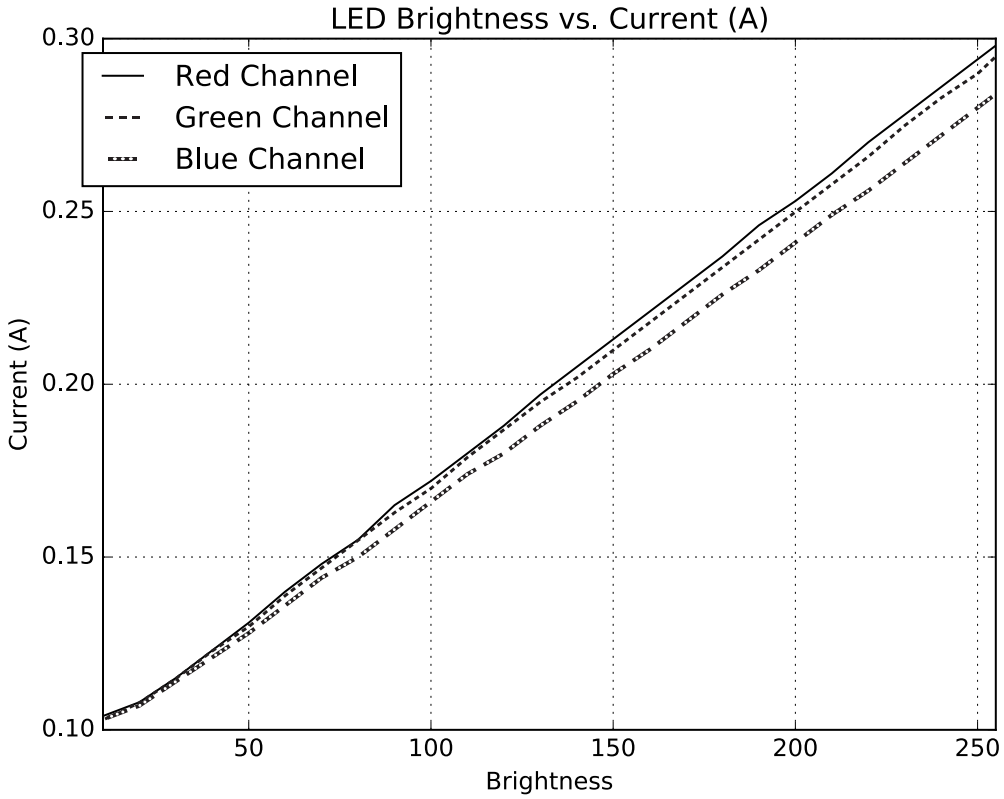

Figure 3.4: Current draw for each LED colour channel in brightness increments of 10.

3.5.3 Analysis

Battery Life

Looking at table 3.1, it is clear that the broadcast rate has little effect on the battery life of the system. The shortest active time, 7.62 hours at 10ms broadcast rate, may be explained by a small increase in power consumption by the WiFi radio due to the high frequency of message output. However, by this logic the longest battery time should result from the largest broadcast rate but, when examining the 10,000 ms entry, one finds the second shortest battery life. Therefore, the variation in battery length when comparing broadcast rates is likely due to some other factor. There are several other factors which could contribute:

1. inconsistencies in the charging of the battery between tests;

2. variation in the undervoltage protection circuit threshold due to temperature

fluctuations, and;

3. reduced battery capacity with continued used.

From this data, no conclusion can be made about this capacity inconsistency however it is quite clear that the broadcast rate of the device is not a major contributing factor.

Observing table 3.2, the total effect of the power consumption of the LEDs is clear. When using any of the LED channels on full, the total time reduces significantly when compared to even the worst case scenario for non-LED operation. Comparing the average of average time active for for all channels, 2.71, and the average for all broadcast rates, 7.99, the use of any channel of LEDs on full brightness reduces the overall time to an estimated 34% of the capacity available with no LEDs on.

Furthermore, from Figure 3.4–compiled from tables A.3, A.4, and A.5–it can be seem that the current consumption is fairly evenly matched at low values and diverges a small amount as the brightness increases. While the divergence looks like it would continue with higher brightness levels, 255 is the highest brightness and so the current level divergence is at a maximum at this level. This current divergence is likely designed into the WS2812B chip in order to attain equal brightness of each LED.

An interesting note can be seen from table A.6 in which the total current draw from all three LED channels activated simultaneously is less than the sum of the three channels separately. This hints at a potential current limit in the system. Because each channel's LEDs track so closely at low current, it is not expected that this current limit will affect the colour mixing of the LEDs but will affect the overall brightness of the device when all LEDs are set to full.

The current draw tables also point to an affirmation of the battery life measured by the previous experiment. This is calculated by taking the total amp hours of the battery–0.8Ah–and dividing by the current. At maximum brightness the total operational time for each channel is estimated to be:

1. Red: 2.68 hours

2. Green: 2.71 hours

3. Blue 2.82 hours

While these values are not exactly identical to those observed in table 3.2, the estimated time is close. The variation in actual time is possibly due to natural variations

in the battery capacity or variation in the undervoltage protection threshold resultant from temperature fluctuations affecting the circuit. That being said, no definitive conclusion can be made about this variation without further tests.

In summary, the power consumption of the radio is negligible, especially when compared to the large current draw of the LEDs. Furthermore, while the battery life of this device is not exceptional, even with one channel of the LEDs on full, the battery life should be effective for prolonged applications, including live performance and interactive installation. While changing or charging the batteries every 3 hours for an installation may be a nuisance, the batteries are removable and chargeable outside of the device. Furthermore, use in an interactive piece does not necessitate full brightness of the LEDs at all times. Therefore, while optimization of the system to increase battery life is a priority, the results of this series of tests show modifications are not strictly necessary since the capacity should be sufficient for a variety of tasks.

Packet Loss

Through looking at table 3.4, there is a clear relationship between the packet difference and the broadcast rate. With shorter broadcast rates, the maximum packet difference increases. A potential reason for this phenomenon is that the protocol responsible for the transmission of OSC messages, UDP, does not include a mechanism for guaranteeing packet delivery and thus by increasing the transmission speed (broadcast rate), the chances of packet loss increases proportionally. In other words, broadcasting a message every 10000ms will result in 1/100 of the number of messages sent with a broadcast rate of 10ms and therefore, if every packet has an equal chance of arrival, the total number of packets lost at a higher transmission rate will also be larger.

Table 3.3, indicating the distribution of packets, may validate this theory since the distribution of packet differences in all cases is close. In other words, the number of lost packets is proportionally similar for each broadcast rate, pointing to the packet losses as outliers in the system and indicative of the transmission method. Broadcast rates 100 and 10000ms had zero packet loss but some instances of completely successful packet transmission are expected when loss has such a small probability. While there is potential for this packet loss to be due to the hardware or software of the system, there is no evidence of this being the case. In other words, this packet loss is likely due to an inherent quality of wireless data transmission.

Table 3.3 also helps to demonstrates the scope of packet loss for the system. Since the average packet difference for each broadcast rate is small, having over 99.9% of packets below the average plus one standard deviation shows that the total number of missed packets is effectively inconsequential. Furthermore, the averages in table 3.4 can be looked at as an indicator of how few packets are lost in the system. An average closer to 1.0 means complete data transmission and the decimal component of the value can be seen as a packet error. This packet error shows how few packets, compared to the total number of packets send, were actually lost.

Fundamentally, this data shows there is packet loss with the system but that this packet loss appears to affect each broadcast rate with nearly the same probability. It is therefore likely that these lost packets are due to the communication protocol and the loss inherent to a wireless system and are not from errors in the system itself. UDP messaging is a trade off: it is a fast communication method because it does not have any mechanism for ensuring packet arrival and so there is a chance that packets will not arrive. Given the context of the system, this packet loss is essentially trivial: the dropping of an occasional packet is not likely to cause a serious malfunction because the system transmits all OSC messages in a single packet at each update interval. This means that a dropped message is likely to be unnoticed from the continuous stream of messages since no corrupted or malformed data is expected to arrive. Therefore, message inconsistency is not an issue with the system. Ultimately, some packet loss is expected in a wireless network and, while not ideal, the actual number of missed packets is not significant.

Broadcast Delay

Looking at table 3.5, it can be seen that the broadcast rate of the system does not match the intended broadcast rate set in the software. This may indicate a minimum delay between the time when data is acquired by the system and time it takes for the system to transmit the information. However, a pattern for this is difficult to determine because the minimum values of the 100, 1000, and 10000 ms broadcast rates are below the set rates. This is likely an error in the data recording process since one would expect these minimums to be no less than 2ms more than the set rate, as observed with the 10 and 25ms rates. Therefore, there is insufficient data to explain the timing inconsistency.

In the context of real-world use, timing consistency is critical. However, despite

the relatively large timing variation, table 3.6 indicates that the vast majority of messages are sent within a tight grouping of time, and because all of the average rates are ±6.8ms of the intended rate, the timing should be considered fairly consistent.

Looking at the previous section, some of these timing inconsistencies may be explained as resultant from packet loss since the time between each sample is transmitted as part of the OSC messages. A dropped packet would mean the timing information is lost and there would appear to be a greater delay between message transmissions. This issue reflects the danger of using the system to measure itself.

Another potential explanation for the timing variation is the timing interactions with other processes, specifically the IMU which can interrupt other processes in the system. The benefit of this interrupt mechanism is a consistent sampling rate–i.e. the data from the IMU is retrieved without interruption at exact intervals. Maintaining a consistent sampling rate is beneficial for acquiring clean and reliable data. This interrupt routine, however, may occasionally interrupt the broadcast of messages which would cause a variation in the timing of broadcasts. Because the interrupt routine and the broadcast timing are independent, and each may vary in total processing time, the interaction of these two processes is indeterminate. A potential fix for this problem is the scaling of the IMU data acquisition frequency proportional to the broadcast rate. This may not be idea but if it reduced variation in timing, a slower sampling rate for the IMU would be optimal over variations in arrival of that data.

While the range of timing differences can be drastic, the variation is likely only a serious determent for scientific data acquisition and in the context of real-time user interface, mild timing variation is not likely to be noticeable. Fundamentally, this is an issue which in future development should be considered a high priority in order to make the system useful and reliable across as many domains as possible. However, currently this issue is not considered a significant problem.

Bandwidth

The data present in section 3.5.2 reveals several significant attributes of the system. The bandwidth, running at an update interval of 25ms, is 41 kbit/s. By knowing the bandwidth of the network, the theoretical number of *dreamIO* development boards on a single network can be calculated.

Given that the ESP8266 is capable of a range of WiFi standard (802.11 b/g/n/e/i) the data rates available range from 11 megabits per second (Mb/s) to 600 Mb/s [53].

The latter is the theoretical limit of the 802.11n standard.

The conversion of kilobits to megabits is a division by a factor of 1000, therefore the bandwidth of a single node in the system is .041 Mb/s. To calculate the theoretical number of devices, equation 3.1 is used:

$$TotalDevices = \frac{TotalNetworkBandwidth}{SingleDeviceBandwidth} \tag{3.1}$$

This gives a theoretical network capacity range of 268 at 11 Mb/s devices to 14,634 at 600 Mb/s. Even on the lower-tier of this spectrum, the number of devices is significant. This estimate only considers data coming from devices and not data sent to devices from a host application. Therefore, assuming a single application sending as much data as received, the total capacity of the network is reduced by half.

Another important consideration is that the data rate of each device is a function of the broadcast rate of the device. If the the broadcast rate is increased, the data rate will increase linearly in proportion. Thus, more devices could be attached to the network if the update rate was reduced.

Another valuable attribute exposed with this data is a concrete measurement of packet broadcast rates (as discussed in section 3.5.3). Given the total number of packets received and the total length of time of the session, the average broadcast rate of the system can be calculated with equation 3.2:

$$AverageBroadcastRate = \frac{TotalTime}{TotalNumberPackets} \tag{3.2}$$

Therefore, the average broadcast rate is calculated as: 29.48 ms. This value is close to the average presented in table 3.5 and assists with the validity of both datasets.

The flaw of this data is that it is only for a single session with a single device. For future research, more Wireshark recordings with variability in number of devices and data rates is likely to yield significant findings.

Fundamentally the number of nodes possible on a network is large and, even at the lowest theoretical values, amounts to a significant potential for large networks. While the effect of multiple hosts and multiple nodes on the integrity of a system is not known, these early findings indicate that hundreds of nodes should be possible. Furthermore, in the event of a network hitting capacity, the use of multiple WiFi networks is a potential solution.

3.6 Future Work

Future iterations of the *dreamIO* platform can be further improved in a number of ways:

First, in terms of software, the firmware can be extended with more feature extraction methods. A greater number of available features will expand the creative potential when designing interactive applications. The firmware can also be extended with a larger variety of input control parameters, including: LED animations, manipulation of the LEDs in different colour spaces (e.g. HSL), and different update modes for data transmission (e.g. event based messaging). Control of these new parameters would be made available through an expanded API.

Second, in terms of hardware, the size of the circuit board can be decreased. This will reduce the overall price of these hardware nodes. A reduction in size will also require the number of onboard LEDs to be reduced which, although negatively impacting light output, will lessen the impact of the LED current draw on battery life. Furthermore, the hardware can be extended to increase the number of integrated sensors and actuators. This may include a vibration motor–to add haptic feedback–and a more sophisticated IMU containing a magnetometer–for increasing the accuracy of yaw, pitch and roll calculation.

3.7 Conclusion

This chapter has explored the development, implementation, and fundamental characteristics of a system for creating interactive art which combines features of TUIs and WSNs with the benefits afforded by application design through a WiFi based API. It has been shown how interactive installation art is limited by current technologies which hinder creative potential through complexities associated with communication. Specifically, that there are few relevant tools for use in interactive art installation and many of these are hindered by scalability and communication limitations.

Through the development of a hardware and software system which leverages WiFi, combined with modern software design techniques–namely the concept of 'application' design utilizing an API–I have shown how these problems can be mediated. By reducing the development overhead–both in terms of cost and time–for interactive designers and artists, the *dreamIO* system increases the accessibility of interactive technologies for a wide variety of users and applications. Furthermore, several ex-

tended usage scenarios have been presented exploring how the system can be utilized beyond the context of visual art.

In the presentation of data acquired from the system, I have shown that, while there is room for improvement in all aspects of the system, there is sufficient battery life, communication integrity, and scalability for use in real-world applications. In effect, the goal of this project was the creation of a tool for use in a wide variety of interactivity contexts with as little limitations as possible. I believe this goal to be accomplished and, given the potential for expansion and further development, both by myself and external parties, any of the current issues of the system are likely to be ameliorated in the future. Furthermore, given the open nature of the project, I believe there is potential for the system to be expanded far beyond the original design intention.

The next chapter will explore the use of the *dreamIO* platform as the core building block for an interactive art installation.

Chapter 4

Transcode

Interactive Digital Art Through Relational Aesthetics and Cybernetics

Transcode: *to convert (language or information) from one form of coded representation to another [49].*

4.1 Introduction

The development of the *dreamIO* platform was initiated to solve some key issues in the design and implementation of digital interactive installation art. This process covered the overarching design principals of the system but does not present any full-fledged examples of use for creating interactive works.

Transcode is an interactive art installation designed with this in mind, solving two problems: first, to act as an example work for the *dreamIO* platform, showcasing the potential of the technology, and; second, of equal importance, to act as a standalone and competent work in the context of visual art.

The platform was designed to be a useful tool for visual artists. By providing an example work I hope to increase adoption by demystify the functionality of the system. Furthermore, it is crucial that the full capabilities of the technology be showcased to reveal its potential. In creating this installation, I validated the necessity of developing the tool by creating a work which would be difficult or impossible without the assisting technology. Therefore, by creating a complete installation, rather than a simple demo, the work is able to accurately represent the value of the *dreamIO* framework. In other words, the technology and the installation must sit as equals since they hold each other

up. Thus a new and authentic installation was required in order for this equality to exist.

The installation utilizes the *dreamIO* platform as the core building block for facilitating interactivity. However, the work also weaves elements of digital signal processing and an interpretation of Visual Music (explored in section 4.5.4) in order to create a new set of experiences. Furthermore, the combination of technology and interactivity of the piece is used to explore connections made possible through digital media. The relationship between the technology, interactivity, and aesthetics of the piece was designed to merge concepts from each domain–computer science and visual art–into a coherent approach to modern digital and technology based visual art practice. This includes utilizing concepts of computation–data, memory, and algorithms–as core components to the form and aesthetics of the piece. By reducing the barrier between the digital and physical world, through utilization of digital interfaces tightly integrated into the installation, I attempt to bridge these worlds in order explore new territory in the domain of visual art. Furthermore, through the study of these two domains, I seek to create conceptual links for digital art theory as well as application of computer science. I believe these conceptual links will help in the generation of new forms of digital art.

4.1.1 Claims

Several claims are made which are validated in this chapter:

1. *Transcode* is a full fledged interactive installation, rather than a demo, and is only made possible through the creation of the *dreamIO* platform. Furthermore, I make the claim that this piece is an attempt to formalize the idea of interactive 'application'–a concept borrowed from software development–within the context of digital art;

2. this piece reflects the goal of unifying multi-modal media art into a coherent experience with an emphasis on accessibility and approachability through interactivity, and;

3. Third, the core ideas driving the piece are made possible through the interaction of knowledge between the domains of computer science and visual arts.

These claims will be validated through presentation of the piece, discussion of the core elements driving the work, and through an analysis utilizing Cybernetics and Relational Aesthetics.

It will be shown how complexity of interactivity can arise from simple actions enabled by small discrete units formed into larger structures. Along with this, the value and necessity of cross-domain study in terms of modern digital art will become apparent. Specifically, how a focus on the acquisition, transformation, and flow of information is a necessity when considering modern digitally based works. Furthermore, I argue that this focus on information is potentially more important than considerations of the physical form of the piece especially when considering the non-physicality of digital media. In other words, because digital media does not reside solely in a physical form, alternative methods of analysis are necessary for understanding these works. While no fundamental conclusions on the totality of digital art are presented, the domain of interactive and digital art is expanded through a multi-domain perspective.

4.1.2 Overview

In section 4.2, the motivations behind the creation of *Transcode* are discussed, including the exploration of the *dreamIO* platform for creating interactive art applications. Next, the design of the physical form of the installation is discussed in section 4.3. Section 4.4 covers the interactivity and software components of the piece. The purposes and core motivations behind the design of the installation are discussed in 4.5, and, finally, the work, and digital art more generally, are discussed utilizing Cybernetics and Relational Aesthetics in section 4.6.

4.2 Motivating Interactive Applications

There are several reasons for the creation of this installation: First, the piece is an exploration of the *dreamIO* development board in an attempt to create a new work which leverages the technology in such a way that the piece would be difficult, if not impossible, to create without this tool. This is by no means an exhaustion of the potential of the *dreamIO* as a tool for creating interactive art but works as a starting point for use of the system. Second, the piece constitutes a new work and an extension of my artistic practice, leveraging the knowledge and understanding acquired through

study at the graduate level. In essence, *Transcode* is a synthesis of the tools, ideas, and processes which I have developed through cross-domain study. In the creation of this work, I attempt to push at and explore the boundaries of digital and new media art through knowledge derived from a combination of computer science and visual arts.

This work was created to further the understanding of digital interactive works–a medium with complex issues and attributes not present in the majority of artistic mediums–within the domain of art. These issues include non-physicality and interactivity, both of which requiring a broad range of cross-domain resources and knowledge to tackle. Through this work, new understandings of the relationship between the physicality of interactive installation, the mechanisms and interface through which the interactivity is enabled, and the non-physical digital or programmatic component, are explored. The piece acts as a study of these principles in order to acquire a greater understanding of the processes behind designing interactivity, and the limitations of the medium, through the construction of a wholly new work. This piece will help to inform my future artworks and the continuous development of the *dreamIO* system.

Finally, the work is a contribution to the field of visual art, designed to be experienced, with little interference from the cognitive loading which led to the piece's development. In other words, while this piece holds significance in terms of theory of form and computation in digital art, fundamentally it is an interactive work created to be experienced. The purpose of the piece, in terms of artistic output, is the cultivation of experiences including exploration and play. This is enabled through the generation of an interactive domain with little connections to users' previous experiences. By enticing users through a pleasant visual aesthetic and immediate interactive feedback, the piece initiate an interactive experience. After this initial intersect, exploration is enabled through the large parameter space controllable by the user. In essence, the form of the piece and immediate interactive feedback are an attempt to lure in users. Once users begin to explore, their experience of the piece begins to unfold. The experiences enable by the piece are fundamentally phenomenological in nature, offering a unique experience for all users through an abstract world of interaction. This functionality and interactivity is totally encapsulated in the work and is fundamentally understood through the experience of interaction–the relationships which emerge between users and the piece–and does not require a didactic explanation.

4.3 Digital Artifacts

Below is a description of the components of the installation.

4.3.1 The Box

The control interface of the installation are small wooden cubes–approximately 73mm to a side. The cubes are made of 5.2mm (1/4") birch plywood. The wood was laser cut with finger joints to allow easy assembly of the cubes. The cubes are hollow on the inside and are designed to mate with the *dreamIO* development board, measuring 60mm per side, allowing the board to sit flush inside of the cube. An online tool was used to design the box [30], allowing the definition of the inside dimensions (60mm to match the *dreamIO* development board) and the joint size.

The box is held together with small cubes of plastic on the inside of each corner. These cubes, manufactured by Bürklin, are 10mm on a side can be seen in Figure 4.1. On the centre of each plastic corner cube is a threaded hole to allow an m3 8mm bolt to fit. On each face of the wooden cube, there are holes 5mm from each corner. These holes allow the 8mm bolts pass through the face and into the plastic cubes which, when tightened, pull the box together. This configuration allows the boxes to be assembled without glue, making it easy to remove the sides of a box and access to the internal components. With the sides removable, the internal components of the PCB, including USB charging port, can be obscured from the user but still maintain accessibility for servicing and charging. Furthermore, the plastic corner cubes, along with holding the case together, allow the PCB to be mounted to the box, via the cubes, while maintaining a space beneath for the battery.

Five of the six sides of each box have symbols laser cut to allow the PCB's LED light to be visible on the outside of the cube. On the inside of the cube, velum is affixed to the back of each face to act as a diffusion for the LEDs. The bottom of each box is left uncut to allow for a space for the battery pack to sit. Furthermore, because all of the LEDs on the PCB face upwards, any cutouts on the bottom face would not yield significant amounts of light to pass. This blank side also helps orient the user by acting as a definitive bottom.

The symbols on each side are a selection of the Linear A [51] alphabet. This alphabet was chosen for three reasons. First, the alphabet is long dead and should therefore be free of any issues of cultural appropriation (in the public domain so to speak). Second, because the symbols look 'magical', which fits the aesthetic goal of

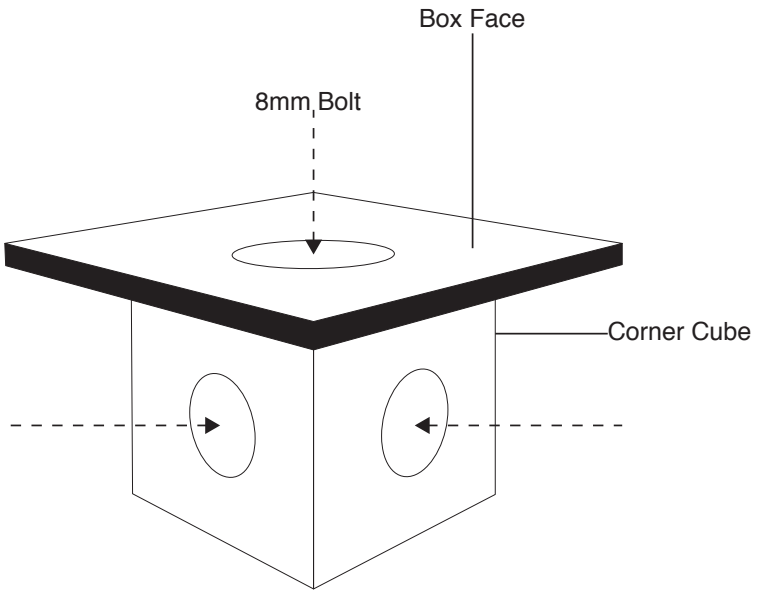

Figure 4.1: Diagram of Bürklin 10mm corner cube in relation to cube faces.

the piece (discussed in section 4.5). And, finally, because the alphabet is available in vector format as a block of the Unicode 9.0 standard [50]. This allowed me to easily transfer the vector files into the CAD software for prototyping and final production. The vector file including the selected symbols can be seen in Figure 4.2.

The specific symbol selection was based on several metrics. First, a subset of these symbols were selected for having stencil characteristics. This primarily meant having no floating points since any floating point would not translate when cut. Next, characters from the subset were chosen for their aesthetic properties based on the underlying goal of 'magical' looking symbols. Specifics of this aesthetic choice are outlined in section 4.5. Finally, from this smaller set, constituting 55 symbols, the artist digitally laid out the patterns in CAD. This set was laser cut with several prototype box faces in order to guage how the symbols worked on the actual form. From this step, a set of 5 symbols were chosen.

When cut into the box sides, these symbols act like icons in a computer environ-

ment, reflecting the digital to physical connection of the piece. This digital-physical relationship is one of the core mechanisms which drives the piece and the symbols help represent it. The symbols are non-representational, in that the symbols carry no previous meaning, and act primarily as a mechanism to help user to differentiate between sides of the cube. Each side of the cube acts as a mechanism for control of digital parameter space. These symbols, therefore, help orient the user in this space by acting as landmarks for physical navigation. Numbers could have been used but I was concern the cubes would too closely resemble dice and thus result in preconditioned interaction (dice throwing). Similarly, if representational images were chosen, the images would come preloaded with associations which would have obstructed the experience of the installation. Furthermore, if common symbols were chosen, the esoteric and pseudo-magical aesthetic of the piece would be lost. This esoteric aesthetic is meant to help entice exploration by users through providing a feeling of mystery and the unknown; something more than they understand but that which they want to know.

4.3.2 Control Interface

The shape of the enclosure is important. The specific size of the case is such that it can easily be manipulated in one hand. Physical manipulation of the cube allows the user to control the installation made possible through the IMU on the *dreamIO* board. The interface is configured such that each side of the cube acts as a separate virtual 'knob'. As a result, rotating a cube with side a facing up can control one system parameter independently from rotating the cube with side b up. Each side, or knob, is assigned to control separate parameters and this assignment is static. Therefore, the symbols on the sides of the cube are locked to specific parameters and act as markers to orient the user in the parameter space. This control is enabled through mapping the sides in software to be represented digitally. This mapping can be seen in table A.2.

4.3.3 Iterative Design and Rapid Prototyping

The physical form of the control interface is a product of an iterative design approach made possible through rapid prototyping. Each phase of the design was rendered digitally and then laser cut with a CNC machine. This allowed for a quick succession of design tightly coupled with interaction with physical manifestations of the design.

Cases of different materials (cardboard and wood), different physical sizes, and side symbols were created in order to hone in on an ideal form for the control interface.

The technologies utilized to prototype the case for the interface were also used for the final production. While this does create some limitations–for example, the CNC laser cannot cut wood thicker than .502mm–this strategy yielded significantly reduced overhead when moving from prototype to production phase. The files used to drive the laser cutter can be scaled allowing the size of the case to be modified easily. These files can also be used with other CNC machines–e.g. mills, water jet cutters, 3D printers–and can therefore be used to make the cases out of materials other than wood.

4.4 The Installation

The work consists of an interactive application written in Max/MSP and controlled by five *dreamIO* nodes. An interactive projection provides the basis for the installation. This visual component is based on early electronically composed images by Ben F. Laposky called Oscillons [29]. These images, created using cathode ray tube (CRT) oscilloscopes, consist of complex curves resultant from the mapping of electronic signals to the screen. This visual phenomenon is replicated digitally in the installation with parameters controllable by the user. The control paradigm is based on the interface of a step sequencer merged with the wireless control made possible with the *dreamIO* framework. Five separate control interfaces allow users to manipulate a large set of parameters wirelessly. The floor plan for the installation can be seen in 4.3. The details of these control parameters and the projection are discussed below.

4.4.1 Visual Display

Laposky created Oscillons using using CRT oscilloscopes to generate complex graphics which were then photographed and processed to make mono-chromatic and colour images [25]. These complex graphics are the product of feeding an oscilloscope two oscillating signals to control the position of a point on the CRT screen. For this configuration, one signal controls the X position of the point while the other controls the Y position. At high frequencies the point on the screen moves faster than the refresh rate of the display and the path of the point becomes a continuous line. By

tuning the frequencies of oscillating signals and changing the phase, complex images are formed. This type of visualization is also called a Lissajous curve and the core signals which generate these forms share many similarities to synthesized music.

This analogue process was emulated digitally in Max/MSP using the Jitter framework and projected on the gallery wall. Four sinusoidal oscillators were created. Two of these oscillators independently control the X and Y position of the visualized point. The other two oscillators are used to modulate the phase of the X-axis and Y-axis oscillators. The frequency of the X-axis oscillator is directly controllable as a parameter while the other three oscillators' frequencies are controlled as a multiplier of the X-axis frequency. In other words, the X-axis frequency is able to be controlled directly and the frequencies of the other three oscillators are some multiple of this value (controlled by the user). This allows the frequencies of all of the oscillators to be linked and creates a coherency in the visual display through these frequency relationships. This coherency would be difficult to maintain with free form control over all oscillator frequencies given that no relationship between frequencies would be guaranteed. Observing the Lissajous example curves in Figure 4.4, the value of linking oscillator frequencies becomes apparent since when oscillators are set to be multiples of the original (X-axis) frequency they can become harmonically linked as whole number ratios. This allows the user to transition between inharmonic–non-whole number multiples–and harmonic relationships–whole number multiples of the base frequency. Furthermore, because the oscillators which drive the visual display are functionally identical to those which could be used to synthesize music, this harmonic linking, which yields smooth curves in the projection, is identical to harmonics in the musical domain. This helps situate the installation in the history of Visual Music which will be discussed later.

4.4.2 Parameter Control

Five *dreamIO* boards are used for control of the visual component of the installation. Each face of the cube provides a different parameter for control. More specifically, the properties of the case housing the circuitry–a six sided object–and data available through the *dreamIO* API–on which side the cube sits is know–are combined to multiply the IMU sensor readings into six different channels of data. To illustrate this configuration, when a cube is on one side and rotated, the code running the installation is able to map that motion to a different parameter than when the cube

is sitting on other sides. Rotation of the cube on each side can be thought of like having separate knob like controls–similar to a volume knob–which are mapped to different parameters of the Lissajous visualization. In practice, only five of the sides are 'active' allowing the sixth side to act as an 'off' or 'mute', creating a parallel for the musical concept of 'silence' in the work.

Four of the cubes–dubbed the step cubes–are configured to act as individual 'voices' for the visual display. These cubes control identical parameters and control of these parameters is shared based on activation (discussed below). The fifth cube acts as a master control allowing for manipulation of parameters unavailable to the other cubes. These parameters affect the entire state of the installation. This is in contrast to the step cubes which directly control the point displayed on the screen. The different parameters are outlined below:

The 'master' cube parameter map (per side):

1. Beats per Minute (BPM) of the sequencer; i.e. the speed at which the sequencer steps.
2. Slew rate of step sequencer; i.e. the transition rate of the parameters from one step to another.
3. Visualization decay; i.e. the length of time it takes the oscilloscope screen to fade.
4. X-axis rotation oscillation frequency multiplier; i.e. the rate at which the visualization rotates around the X-axis.
5. Y-axis rotation oscillation frequency multiplier; i.e. the rate at which the visualization rotates around the Y-axis.
6. The installation turns off. Started when turned to any other side.

'Step' cubes parameter map (per side):

1. Main oscillator frequency–controls the X-axis oscillation
2. Y-axis oscillator frequency–multiple of main oscillator frequency.
3. Main oscillator modulation frequency–multiple of main oscillator frequency.

4. Y-axis oscillator modulation frequency–multiple of main oscillator frequency.
5. Projection point size; i.e. the size of the moving dot.
6. Mute, turns the cube off in the sequence causing all parameters to be set to zero.

A sixth parameter is controlled by the step cubes: each side of a cube is mapped to a specific colour which is displayed on a cube via its LEDs. This same colour is also used to define the colour of the Lissajous projection. When a step cube is activated, its LEDs turn on and change to this colour along with the projection. The goal of this mechanic is to create an obvious connection between user action and the reaction of the installation for immediate feedback. Step cubes are activated through manipulation and remain active until another cube is touched.

4.4.3 Step Sequencer

The interactivity of the installation is based on a step sequencer or drum machine interface. Four of the cubes in the installation act as steps in a sequence, activating one at a time in series. When a step is activated, the associated cube lights up and the parameters of the cube are transmitted to the program and visualized onscreen.

When a step cube is activated the sequencer pauses, allowing a user to directly manipulate the Lissajous and therefore able to directly perceive the affect of their actions on the display. This cube maintains control until one of three events occurs:

- another step cube is activated (moved)–this cube takes control of the installation
- a master control cube is moved–causes the step sequence to begin again
- no cube is activated for five minutes–also causes the step sequence to begin

This interactivity is made possible through the *dreamIO* API motion sensing capabilities. i.e. the API allows the *Transcode* software to know if a cube is currently being touched. This type of activation emulates computer operating systems' mechanism of focus–clicking on a window activates it allowing the user to interact with its contents.

4.4.4 User Feedback

Because the parameters controlled by the cubes are abstract it can be difficult for a user to see the direct effect they are having on the installation. To counter this ambiguity, several ways of providing feedback to users are present in the work.

First, while the IMU can provide fine control over the parameter space, many of the parameter controls have been quantized to make the control obvious to the user. This quantization creates quick jumps between different parameter values rather than smooth transitions as the user manipulates the interface. The parallel for this in music is a theremin compared to a piano. A theremin provides continuous control over the frequency of musical note produced while a piano can only produce 88 discrete notes. This more rigid control structure allows the end user to see significant difference (jumps) in output from small manipulations of the control interface and therefore creates a stronger coupling between action and reaction in the piece.

The LEDs onboard the control cubes are used to display several different sets of information to users. These parameters are persistent and so the symbols on the sides of the cubes are meant to help the user orient themselves in the parameter space. In order to emphasize this, each cube side is assigned a static colour which displays on the LEDs when a cube laying on that side. e.g. side A will always be red for all cubes. Because some of the control parameters for the projection are quite opaque, a mechanism for directly representing control parameter values was created: on each cube, the current parameter value is represented to the viewer via the onboard LEDs as an oscillation in light intensity. The oscillation, perceived as a blinking to the user, has a frequency proportional to the control parameter's level. As a result, as the user turns a cube clockwise, and the parameter being controlled is increased, the lights in the cube begin to flash more rapidly, as the cube is rotated counter-clockwise the pulsing of the lights slows down. These display mechanisms amount to a lexicon or vocabulary which can be learned by the user through interaction.

While the interactivity of this piece is not meant to be exceedingly difficult, the number of parameters and their abstract quality does result in a steep learning curve. This difficulty is not detrimental because the installation is purposefully opaque and is meant to be explored. Beyond the direct effects of the user's actions, there are few clues to guide these actions. This is because the fundamental goal of the piece is to create a new experience which users want to explore without forcing a prescribed method. Learning how each side of the control interfaces affects the piece is a hopeful

outcome but each user is expected to explore the piece in their own way. Therefore, the symbols on the cubes are only markers to help orient the users and do not provide any detail on the interaction. In essence, the structure of the piece is such that users feel attracted to it but with little context to define their interactive experience. The user should feel free to play.

4.4.5 Iteration

On August 27, 2016 a 'beta' version of the installation was installed at the G++ Gallery as part of the Integrate Arts Festival held in Victoria, BC. The piece was only up for one evening and operated within a space with multiple competing elements. The venue became quite the party and there were other other interactive pieces in the room. While the setting is significantly different from the final gallery version of the piece this experiment lead to many helpful insights for improving the overall experience.

From my observations, the piece was well received and garnered continuous attention for the entire event. However, there is some indication that the nuance of the controls was more complicated than many people had patience for. As a result, at the beginning of the night I provided several informal tutorials on operating the piece. After a few of these tutorials, the information necessary for operation seemed to spread through communication between users and continuous interaction throughout the night. While this level of multi-user communication for coordinated action is not likely reflective of a gallery setting it does show that multi-user interaction and collaborate is enabled by the piece. As the result of this experience I decided to alter the parameter controls of the piece to have a smaller and more discrete range of values (as discussed above).

Because of a bug in the installation code, there were times when the piece would not react to users' actions. As a result I observed several users give up after only a few moments of use. It was clear this was because they could not tell if any of their actions were having a direct effect. This is a clear indication that instant and direct feedback must be maintained to sustain interest by users. The details of the need for this direct feedback are discussed in section 4.5.2.

Finally, while the installation was only up for an evening, I observed several users spend significant amounts of time exploring the piece. I cannot speculate on how long any of these users would spend if given the opportunity but this interest may indicate

that the interactive elements of the piece are complex enough to maintain prolonged interested.

4.5 Purposes

4.5.1 Physical Form

The *dreamIO* platform was designed to be minimally invasive in the influence of the art created with it. However, the system cannot be totally without influence given the physical nature of the interface. Specifically, the circuit which is integral to the system has physical dimensions which cannot be ignored.

Being the first installation using this system, rather than attempting to create structures to hide the circuits, I decided to design around the physical form. As a result, the size of the cubes is emergent from the choice of materials matched with the physical dimensions of the *dreamIO* PCB. The internal dimensions of the cube are 60mm x 60mm x 60mm, the minimum dimension possible to allow the PCB to fit inside the case. The resultant external dimensions are approximately 70mm x 70mm x 70mm. Fortunately, this yielded dimensions which are very friendly in terms of handling in one's hands.

The cube shape was chosen for its logical equality for all sides. The goal of the installation design is to leverage the technological underpinning to create the necessary demarcation of parameter space. In other words, because the cubes are an interface for a digital space, the different sides of the cube are meant to orient the user in this digital space. When a cube is changed to a different side, the user is navigating to a different parameter and the images or icons on the sides are meant to help the user orient themselves and express this change in control. The icons are statically linked to their parameter space and acquire meaning to users through the course of interaction.

Approachable

This piece was designed to be approachable in order to maximize contact with the installation. New technologies can be uncomfortable and may prevent users from interacting with the piece. Approachability in this context means lowering the barrier to entry through masking the underlying technology. By making the piece colourful

and relying on an organic aesthetic of wood, the piece reduces the generally sterile feeling of the highly digital to something which feels graspable and approachable.

While the physical form of the boxes is emergent from the dimensions of the circuit, the resultant size is near ideal for approachability. Larger boxes would be more sculptural but, given the context of gallery art which is not always supposed to be touched, this format would make the cubes less approachable.

Approachability is important because the interaction element is essential to the piece and so any barrier to this needs to be reduced. Furthermore, it is through the embodied use of technology that it is naturalized and legitimized [25]. To embody computation, through giving it an approachable form, the underlying technology can become more accessible and useful in the future. In this way, one purpose of the piece is to act as a vanguard for emerging embodied computation.

Despite this goal of approachability, there are still difficulties to overcome to lower the barriers which may prevent interaction by users. This piece was created to be presented in a gallery setting and as a result must mediate a problem: most art in galleries is not meant to be touched. Barring any signage, which would be aesthetically uninteresting and is not guaranteed to work, the piece must somehow present itself as 'touchable'. The approachability of the work is meant to lower this barrier. Furthermore, the layout of the piece, with the placement of the control cubes on waist height plinths, is meant to convey this permission to viewers. However, there are still issues with this setup which may need further development in order to reduce this barrier and enable more participation. Without effort, there is the possibility that the majority of viewers will think the piece is a video work accompanied by sequenced sculptures rather than a fully interactive work.

Accessible

Accessibility in the context of this piece is tied to the aforementioned concept of approachability. While there is nuance to the control schema which can allow users to learn and practice usage towards proficiency, the piece is setup to have immediate positive feedback for users where their actions cause an observable change in the behavior of the piece. This design is intentional to motivate users to play and not be encumbered by the highly technical nature of installation. The hope is that an immediate accessibility will lead to prolonged participation and therefore an extended experience with the piece. In other words, accessibility is bait which draws the user

in and keeps them interested and motivated to explore.

The interactivity design of the piece models a digital music interface, specifically attempting to create an equivalent visual metaphor of a sequencer or drum machine. A drum machine is a simple interface for creating complex expression in music with very little training and while not all users are expected to see the parallel, the simple interface allows for complex expression in the visual domain. This interface also helps to situate the installation in the history of *Visual Music* (discussed below).

By reducing the barrier to entry, there is potential for novice users to achieve high-level individual and collaborative interaction [15]. Continuing with the musical analogy, this high-level interaction is similar to group improvisation which is normally only possible with musicians having a minimum proficiency. Because this installation is altogether new, virtuoso are unlikely and so by designing with accessibility in mind all users should be able to immediately achieve meaningful experiences with the piece or at minimum feel comfortable experimenting and exploring.

Duchamp said "Art is a game between all people of all periods." (as cited in [10]) and *Transcode* sits in this philosophy by motivating the interaction through playability. Furthermore, this piece attempts to bridge concepts from the past (visual music, electronic music, early electronic art) in order to create a logical and meaningful history. Since the piece has it's own 'memory'–as in, it retains its state between different users–it can be thought of as a literal game between people of different periods, albeit short periods.

Pseudo-Magical

The cubes were purposefully designed to appear otherworldly and magical. The inspiration for this is a play on the idea put forth by Clarke that "Any sufficiently advanced tech is indistinguishable from magic."[13] By wrapping the computer in a magical aesthetic, a cognitive divide is created: The user knows these objects are not magic and are actually computers but the cubes are not a form of a computer which the user has seen before. Because some people have an aversion to the highly technical, the intention of this cognitive divide is to reinforce the approachability and accessibility of the installation (as mentioned above) and lower the barrier of entry for play and interaction.

This magical aesthetic is defined by the handcrafted look of the wooden cases juxtaposed with the quality of light radiated from the LEDs. The wood is a natural

substance but the symbols on the cube emit a non-natural glow. These symbols are meant to appear runic and suggest to the user that the object was constructed for a purpose. In other words, someone made this object and it has power. This design is intended to lure the user into their first interaction. Without symbols which look purposeful–such as abstract geometric images–the user may not consider the object a device to be used. This is especially important since in most cases items in a gallery are not supposed to be touched and it can be difficult to tacitly convey the permission to do otherwise. This facade is also intended to be aesthetically pleasing. By placing the circuitry–cold and calculated–into a wooden case, and diffusing the LEDs, the work is made organic and warm.

Finally, while not actually magic, I find when deep into the experience of making music or art, either individually or in collaboration, a transcendental or altered state of consciousness can occur. The technological or highly technical nature of tools are not a barrier to this experience, as made apparent by artists Beck, Siegel, and Paik [25]. The design of this piece is to link the assumed aesthetics of the mystical–wood and handcrafted–with modern technologies. Furthermore, according to [25], "...ongoing developments and innovations in science and technology are preconditions for transcendental thought and desires..." With this in mind, the aesthetic is also an attempt to help facilitate a meaningful relationship between the users and the technology, helping inspire innovation and creativity through demonstrating the possibilities of modern technology.

4.5.2 Interactive Experience

The design of the interactivity for *Transcode* heavily leverages the capabilities enabled by the *dreamIO* framework. Fundamentally, the hardware nodes allowed for the creation of a direct and physical connection between users and the interactivity of the piece. By using boxes which are manageable in a single hand, a range of movement was constructed with minimal constraint on the user. While only certain specific motions create direct results from the piece any physical movement of the nodes will cause some change. It is this direct responsiveness through haptic engagement which comprises the primary interactivity of the piece.

This interactivity was fundamentally constructed to allow for the act of play. The aesthetics elements were designed to minimize the external reference points for the user in order to reduce preconceptions about how the piece works. By doing so, I

hope to maximize exploration by users through motivating their curiosity. This would ideally place the user in a sort of 'pure' experience in which they are focused solely on exploring the installation.

An important element for maintaining this relationship between the user and the interactivity is the immediate feedback which is enabled by the system. This allows essentially any manipulation of the interface to have a result reflected on the installation. I believe this can act as a motivation for maintaining the interest of the user. Related to this issue, Schloss argues that performance in computer music concerts can be hindered as cause and effect are made less transparent through modern musical interfaces:

> This means that we can go so far beyond the usual cause-and-effect relationship between performer and instrument that it seems like magic. Magic is great; too much magic is fatal.

The observer's perspective of the performance must therefore be considered when creating a composition in order to assure the performance is convincing and effective [44]. In the same vein, *Transcode* attempts to maintain the relationship of cause and effect in the piece, through immediate feedback, in an attempt to maintain an active relationship with those interacting with it. Without this consideration, the interactivity of the piece may completely escape the participant as they may be unable to orient themselves in the virtual space without intelligible feedback.

The abstract nature of the visual display is part of this attempt to reduce external references in the piece. The abstract and nuanced nature of the visual component is coupled with the constant and continuous feedback is an attempt to replicate the play and exploration which comes from learning a musical instrument. I believe that although the number of variables in the piece is significant, given enough time a user could become proficient at 'playing' the piece like any musical instrument. Once again, this is the reason the piece can be understood as Visual Music. Keeping with this, the purpose of using a visual abstraction instead of sound is to create an egalitarian potential for the piece. If sound based musical elements were used then people without musical backgrounds may feel impeded and might not explore. Instead, the playful elements of music were wrapped into a new visual experience which can be approached by anyone. The piece leverages these elements in order to create the potential for a meaningful experience.

4.5.3 Design Prototypes

The design of the installation moved through several prototypes to arrive at the final form. The prototypes were created rapidly, aided by access to a computer numerical control (CNC) laser cutter. This tool allowed for the case to be designed in computer-aided design (CAD) software and then produced physically within a matter of hours. The short timespan of design to production allowed for rapid prototyping using different materials (cardboard and wood) and different sizes. This process allowed me to explore the physicality of these digitally designed prototypes with little delay between iterations. Once a case was produced, the form was observed and the design improved to be cut again.

Physically working with the case allowed for a stronger understanding of how to improve the design to best fit the goals of the installation. In working with the materials directly–rather than having the forms manufactured professionally–the limitations, effectiveness, and potential issues became apparent and allowed me, as the artist, to create the final physical form with a focus on functionality.

The first prototype was cut from cardboard (Figure 4.5) and was used with an early prototype of the *dreamIO* framework. This form was overly large and helped to define the size of subsequent forms.

Next, the first wooden forms were created with dimensions based on the final *dreamIO* PCBs (Figure 4.6). This form was the first to incorporate the complex side icons and to explore the joinery for joining together the sides. This form accounts for the broad strokes of the design and solidified the dimensions of the case–it was immediately apparent that the size was very comfortable in one's hands.

The last few forms were essential to dialing in the sizing of the wood joinery, the exact dimensions of the sides–to yield the best fit–as well as finalization of side icons. This stage also included testing different stains and finishes. The final form of the case can be seen in Figure 4.7.

The significance of this design methodology is the ability to tune the end form to meets the criteria of the installation. In other words, this iterative process allowed for the generation of optimal form in a short time span. Furthermore, this design methodology is reflective of the goals of the installation itself–a close coupling between digital and physical–apparent in the back and forth movement between digital and physical domains.

In terms of software design, early versions of the *dreamIO* firmware allowed for

the creation of many of the core code blocks necessary for creating the installation. A Max/MSP object was created first which acts as a mechanism for interfacing with the framework. This allowed for exploration and experimentation with early forms of the *dreamIO* firmware. The ability to work with early forms of the firmware helped to guide the design of the installation and, more importantly, helped to define which features of the firmware API to create. In other words, the software for *Transcode* and *dreamIO* co-evolved through this process.

4.5.4 Visual Music

The concept of Visual Music may originate from an obsession with synesthesia arising in the early 20th century. During this time, abstract painters such as Wassily Kandinsky and Frantiŝek Kupka utilized the formal abstract structure of musical composition as a basis for abstract visual art. Exploration of musical ideas in the visual domain resulted in a variety of strategies, including the application of formal and compositional elements of music into the visual domain as well as exploration of correlations between music theory–pitch, scale, rhythm–and the colour spectrum. The latter resulted in the creation of 'colour organs'–devices which projected coloured light and could be played like a musical instrument. The time-based nature of music also led to the creation of visual media with a focus on time, including abstract film and animation, such as Viking Eggeling's *Symphonie Diagonale* (1924), as well as photographs which focus on subject matter which varies with time–such as Alfred Stiglitz's *Songs of the Sky* (1923/1927/1929) [11]. Because of the immense variety of works and mediums which fit under the concept of visual music, it may be best to consider visual music as the entire domain in which music and visual art intersect.

Transcode operates as visual music in three ways: First, the projection component is situated directly in the lineage of visual music, operating as a visual representation of signals which are driven by fundamentally musical concepts, as well as the physical interface emulating a drum machine. While related, the projection is less in the vein of video art and more closely linked to music, utilizing harmony and dissonance of periodic signals, manipulated through time via the physical interface, and mapped to a corresponding visual space.

Second, the work operates utilizing time as a primary component. Fundamentally, the piece is experienced through time and the core mechanisms which drive the piece are time based–the oscillators controlling the visualization are time varying signals

and the step-sequencer component of the work operates like the ticking hand of a clock. Time is not always a primary component in visual works and this emphasis can be seen as reflecting and embodying the time-based nature of music within the visual domain.

Third, and finally, the piece operates as an allographic work, rather than an autographic work:

> An artist creates an autographic work in a singular act of creation, such as painting or novel. In contrast, creation of the 'allographic' artwork is two tiered. In an allographic art form, like music composition, a written score or notation system forms the first stage of production and the performance of the music forms the second stage of production, or the end product [25].

The interactivity of *Transcode* is such that the underlying algorithm operates as a sort of score with the piece unfolding through time via the performance of interaction. In essence, the allographic nature of the piece creates a strong tie to musical performance and improvisation but within the visual domain. This is made possible through a computational view of signals and interface. More specifically, computational representations of signals can be mapped to control a wide variety of parameters. In other words, a digital signal can be re-contextualized to affect other components of a system with few constraints. The title of the work–*Transcode*–is in reference to the operation of converting information from one coded representation to another[49]. In a digital work, the boundary between visual and auditory is blurred significantly since both types of signal are constituted by the same 'material', bits of information, and are therefore interchangeable[11]. This work was created with an intimate awareness of this new relationship between media which emerges from encoded digital signals. The piece explores this new relationship between information once rendered to the same plane. It is this permeability of signals in the digital domain which allows for the explicit cross-modality, making the work visual music.

4.6 Analysis

Because of the interdisciplinary nature of *Transcode*, two seemingly disparate ideas, each relating more strongly to one domain over the other, are brought together to analyze the work. It will be show how these two modes of thought, Relational Aesthetics and Cybernetics, can work together to situate the piece in both domains, acting as a

bridge. Furthermore, I will discuss how these two modes of thought can be applied to digital art more generally.

4.6.1 Relational Aesthetics

Relational Aesthetics is a theory first described by art critic Nicolas Bourriaud [10]. It is not a theory of art but instead a theory of form. Instead of focusing on the artwork itself, the form is considered an emergent property of the resultant relationship between the artwork, the artist, and the observers or participants with the work. The art emerges from the context created by the artist. This context does not necessitate physical artifacts in order to be constructed. For example, from the perspective of Relational Aesthetic, even a simple conversation can be considered art even though there is no physical artifact constructed for the piece.

This theory can be re-contextualized as a tool for analyzing digital art where the abstract nature of computation leads to the lack of physicality of the work. Digital art–interactive or static–is made up of data and does not have a physical form. The mechanisms for storing, retrieving, and viewing piece (the data) are not themselves the piece. In this context a digital work is without physical form and becomes difficult to discuss using notions of art predicated on physicality.

As a concrete example: consider a digital graphic presented in a gallery on a screen. The graphic is enabled by the screen and, while it must be considered in terms of presentation, the screen is not the art piece. The closest parallel in a more traditional piece is the frame around a painting. Again, the choice of frame is important but the frame is not the painting just as the computer running an installation is not the piece.

Interactive digital works are even more difficult to pin down since the they require viewer participation. The viewer, therefore, become part of the piece. Relational Aesthetics points to the ways in which the elements of the work interact as the form itself. Again, the physical components which make up an interactive work are there to enable the piece but are not the piece itself. Any of the components could be swapped out for an equivalent part and the piece would be the same. The same cannot be said about the interactivity since any change in the specific interactive elements would change the relationship constructed with the viewer. In other words, the interactivity of the piece is a mechanism for enabling relationships with the viewer and any change to this functionality would be a fundamental change to the form

of the piece. Furthermore, beyond the physical aesthetic properties of the piece, the relationship formed between viewers and the work generate the aesthetics of the piece. Each viewer's relationship is likely to be unique as it is formed through the intersection of their personality with the interactivity of the work. Thus, the Relational Aesthetics of this piece are these emergent experiences enabled by the preprogrammed interactivity of the work.

In this context Relational Aesthetics can be a helpful lens when considering interactive digital works when the form of an interactive piece is defined by its interactivity and not its physicality. This is especially pertinent given that some digital works are composed entirely of data.

4.6.2 Cybernetics

Cybernetics is an approach which places importance of transmission of information above the mechanism through which the information is transmitted. Through this perspective, the medium through which a message is transmitted can vary–e.g. electrical, mechanical, biological–without affecting the content of the message. How the information is transmitted is invariant to the transmission method [52]. The same information or message can be represented in a variety of ways, in a variety of mediums, and still be transmitted. Therefore, the method of transmission is less of a consideration than the information transmitted.

The second important component to Cybernetics is the concept of feedback in which the output of a system–the information or data–is fed back into itself, allowing it to adapt its processes. A simple example of feedback in a Cybernetic system is a limiter switch on a CNC machine which is activated when the machine arm hits the limit of its range therefore preventing the arm from moving past its limit. This switch acts as a feedback mechanism allowing the system to augment the input signal–which may say to continue moving the arm–based on real measurement of the state of the machine. Feedback allows a system to dynamically adapt to changes via this closed loop mechanism, through circular and dynamic exchanges of data [25].

This way of viewing has had a significant effect on computation, allowing for the notion of abstraction and portability: the same program can be used on multiple sets of hardware and the end result should be the same. The underlying hardware could even use different mechanisms for computation–mechanical switches, gears, vacuum tubes–and still yield the same results. Cybernetics is the application of

information theory in order to study and analyze all systems in terms of messages or signals rather than their underlying physical mechanisms. This allows machine systems to be flexible, nonlinear, and dynamic [25]. Cybernetics is, therefore, useful for understanding interactive digital works which are constituted by an exchange of data, in a feedback loop, between the different parties involved.

4.6.3 Synthesis

In terms of *Transcode*, and interactive works more generally, Relational Aesthetics can be used to understand interactivity as the underlying form, enabled through the physical interface of the work. While interactive works require some physicality–a physical form or mechanism through which to interact–their physicality is secondary to the function of its interactivity and so when viewed as relational art, the form:

> ...is not the simple secondary effects of a composition... but the principle forms, gestures... The contemporary artwork's form is spread out from its material form: it is a linking of elements, a principle of dynamic agglutination.[10]

Any art which uses code as it's primary medium must utilize a theory of art which does not rely on physicality as its basis. Therefore, Relational Aesthetics is a useful tool when dealing with digital works which fundamentally rely on abstraction via computation. The elements of the art, the underlying code, are conceptual units brought together to create structure, activated by the 'beholder-manipulator' and only made real through human interaction [10]. The underlying technology acts only as a conduit for this experience rather than being the form itself. Without a theory which reduces emphasis of an artwork as a physical artifact, interactive and digital works cannot be considered art, and while Relational Aesthetics may not be the only useful theory for this problem, its emphasis on relationships makes it highly suitable for defining interactive works.

Cybernetics fits nicely with Relational Aesthetics given both have an emphasis on information exchange rather than physical implementation. From a Cybernetics standpoint, an artwork is a system emergent from a process which includes the artist, the user, and the physical hardware of the piece closely tied in a semantic relationship [12]. By combining these two modes of thought, interactive digital works can be seen as constituted by the flow of information between parties. The form of the piece

is this exchange of information–the relationships dynamically produced between the interactive system and the users–and not the physical mechanisms which act only as mechanisms for this exchange. Important to note is in the combination of these two ways of viewing, the interactivity of the piece is a result of this communication and exchange between the users and the system's algorithm, which tacitly places agency in the algorithm by making it an equal actor in the work. The form of the work emerges from the interaction of the users and the algorithm, giving equal importance to each agent. Even if the algorithm is extremely simple and unintelligent, feedback allows for complex behavior to emerge. Without either agent involved in the piece, without users or algorithm, an interactive work is devoid of form.

Transcode is an attempt to focus on inter-human relationships mediated through technology. The underlying physical mechanisms which drive the piece can be swapped out–the computer which drives the piece, the cubes, the projector, can all be independently replaced with essentially zero consequence. While the physical aesthetic properties of this piece are important to how these relationships are forged, they are secondary or subservient to the relationship of interactivity. It is the underlying algorithm (which can also be implemented in other programming languages) and the way in which users interact with the algorithm that generates the form of the piece. Users are able to explore and create a relationship, themselves becoming part of the algorithm through feedback and exchange with the system. In other words, users become part of the mechanism for transmitting information back to the algorithm and are thus inseparable from it. This exchange is made through a relationship between other actors in the system–users co-located in space and time or even previous users of the system. Therefore, the piece can be best understood through the dual lenses of Cybernetics and Relational Aesthetics as having form emergent from the passage of information between different agents in the system–both machine and human–through negotiation and dialog. Furthermore, these two modes of thought can be utilized to better understand the abstract and intangible nature of modern digital works, which lack form but convey information. These works which do not reside in the material but through relations [25]. The changing landscape of art, related to technological developments, is no longer bound to the physical. While not strictly technologically motivated, Relational Aesthetics helps to reorient artistic practice–especially digitally driven works–into a context of message rather than mechanism; content rather than form. Because Relational Aesthetics emphasizes social relations–not necessarily the digital-computational–Cybernetics has value in bridging the technological and the so-

cial worlds together, again through emphasis on information transmission rather than explicit physical form. In essence, *Transcode*, and other digital works, are primarily about communication with their physical form acting as a medium through which to communicate.

Using this dual lens I hypothesis that future interactive works can be designed to maximize these meaningful exchanges between actors in the system. This optimization can be thought of as an attempt to increase the flow of information between actors. This is not to say the best art is that which panders to the largest crowed. Rather, a focus on elements of interactivity which enable a continuous and valuable exchange of information will yield the strongest relationships. Furthermore, I believe the form of interactive digital works can best be discussed in terms of this meaningful exchange and the characteristics of the relationships enabled through the experience.

4.7 Conclusion

This chapter has explored the creation of an interactive art application, *Transcode*, which was made possible through the *dreamIO* platform. This work was analyzed both as an example interactive 'application' to demonstrate the creative potential of the *dreamIO* technology and as a fully formed installation. The purposes motivating design choice of the piece were presented and the development of the piece, both digitally and physically, was explored in depth pointing to a synthesis of principals from the physical and digital domain. This synthesis was shown as an exploration of the form resultant from the interaction of the physicality of the piece and the parallel digital space. This experience of exploration was pointed to as the primary goal of the interactive experience provided by the piece.

Next, the lineage of the installation, situated in the history of both visual art and computer science was examined through a linking with Visual Music. The history of this mode of artistic expression was presented as an early example of bridging between technology and visual art. The influence of this history in the design of *Transcode* was made clear given its focus on time and cross-modality. These ideas resonate heavily with the core motivations underpinning of the piece.

Finally, the installation was analyzed utilizing two modes of thought–Relational Aesthetics and Cybernetics–chosen because each originates from different domains, visual arts and computer science respectively, but share a fundamental similarity. This similarity is a focus on exchange of information (relationships) rather than physical

form. As such, I argue that information, the core focus of Cybernetics, can be used as a bridge between these ideas. This synthesis was pointed to not only as a tool for examining *Transcode* but also as a mechanism for understanding digital artworks, which are inherently non-physical, within the context of visual art.

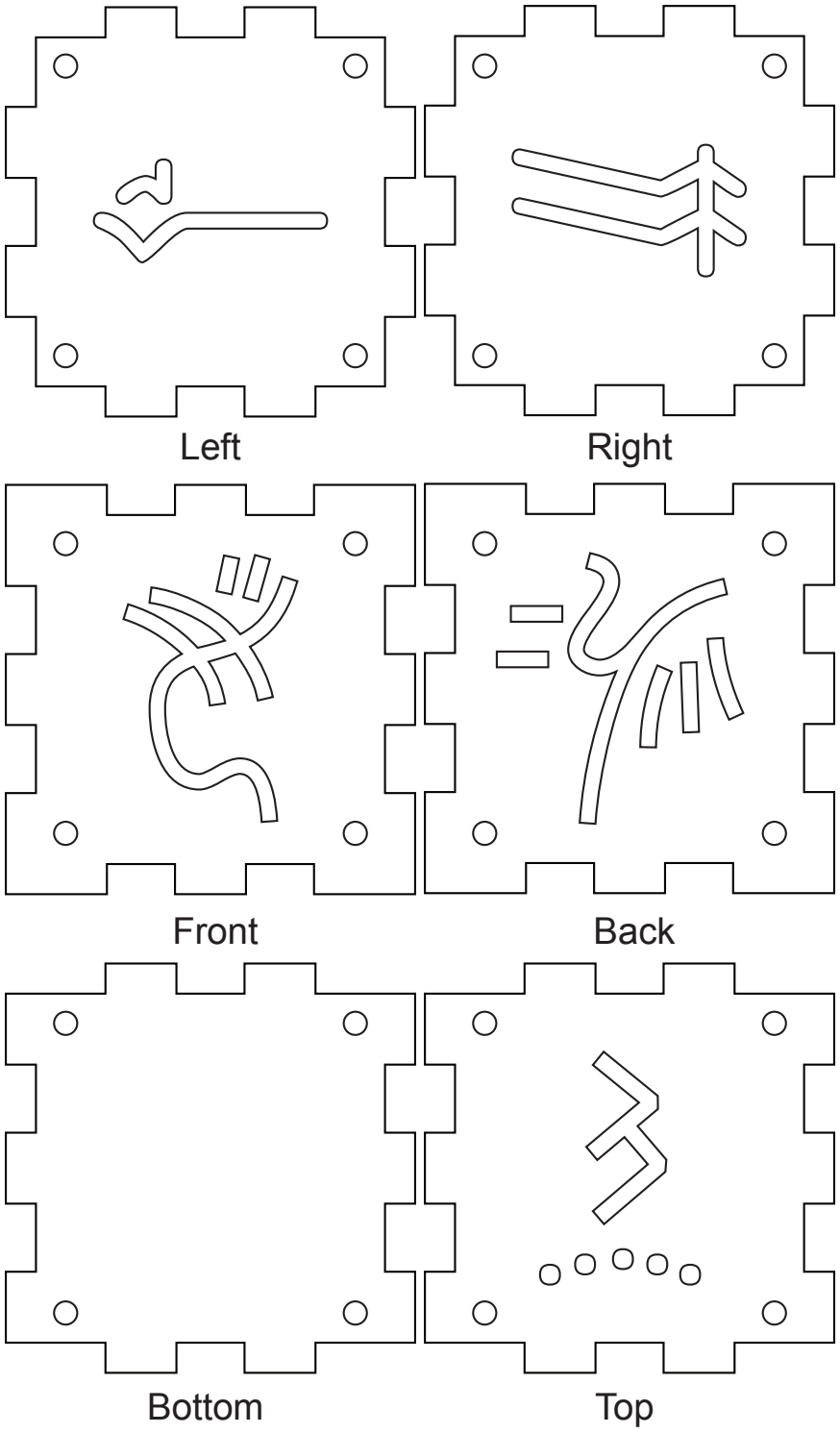

Figure 4.2: Illustration of Transcode case showcasing the symbols selected from the Linear A alphabet. This image is the same vector file used to laser cut the cases. The internal representations of the case sides can be seen in table A.2.

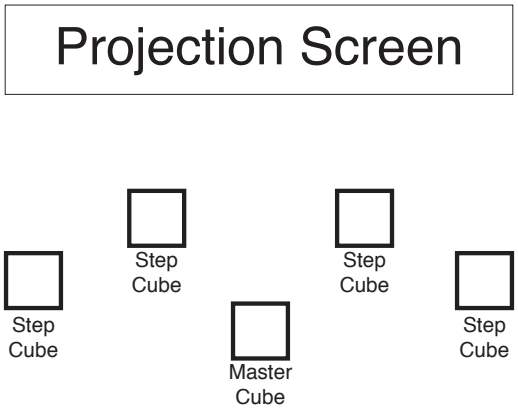

Figure 4.3: Gallery floor plan (top-down view). Cubes are arranged in a semi-circle on individual plinths. The plinths are away from the wall allowing user the ability move around the space and manipulate the cubes while having the projection in view.

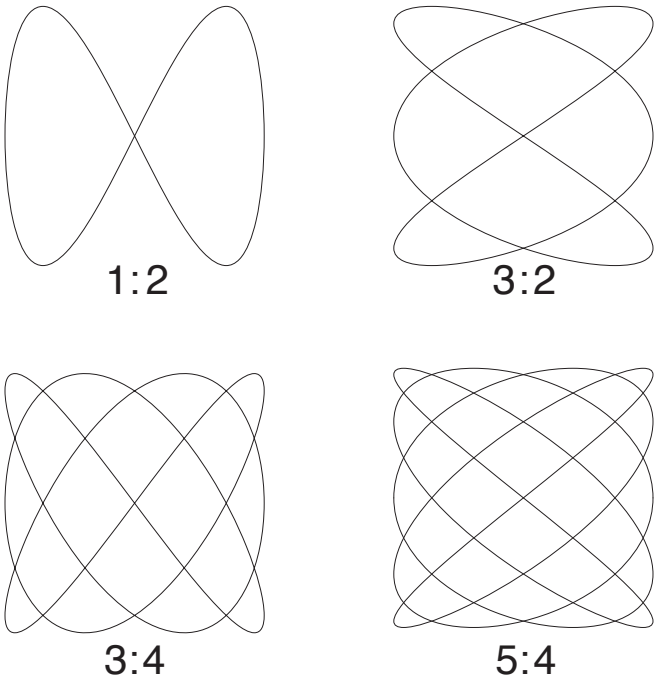

Figure 4.4: Example Lissajous Curves created from different sinusoidal oscillator frequency ratios. Below each figure is the ratio of frequencies for generating the curve.

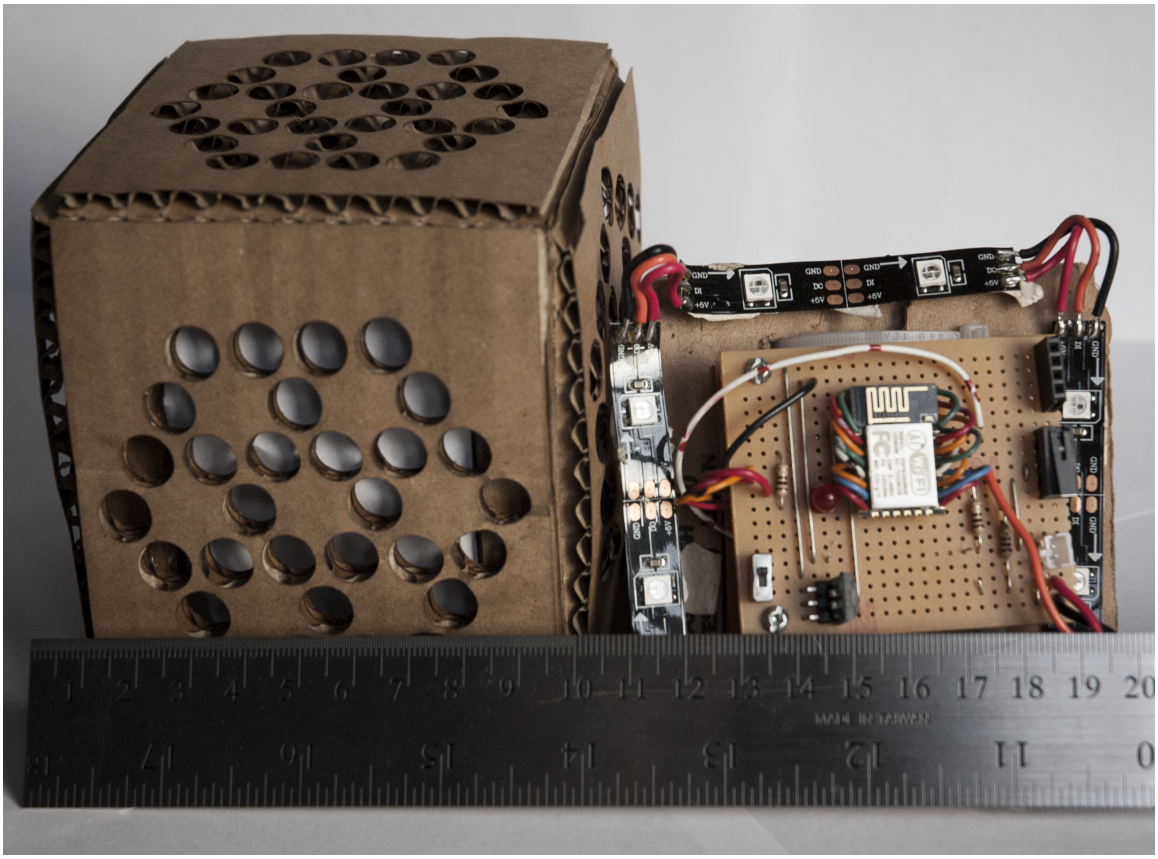

Figure 4.5: Cardboard prototype *Transcode* case next to prototype of *dreamIO* circuit. Ruler for scale.

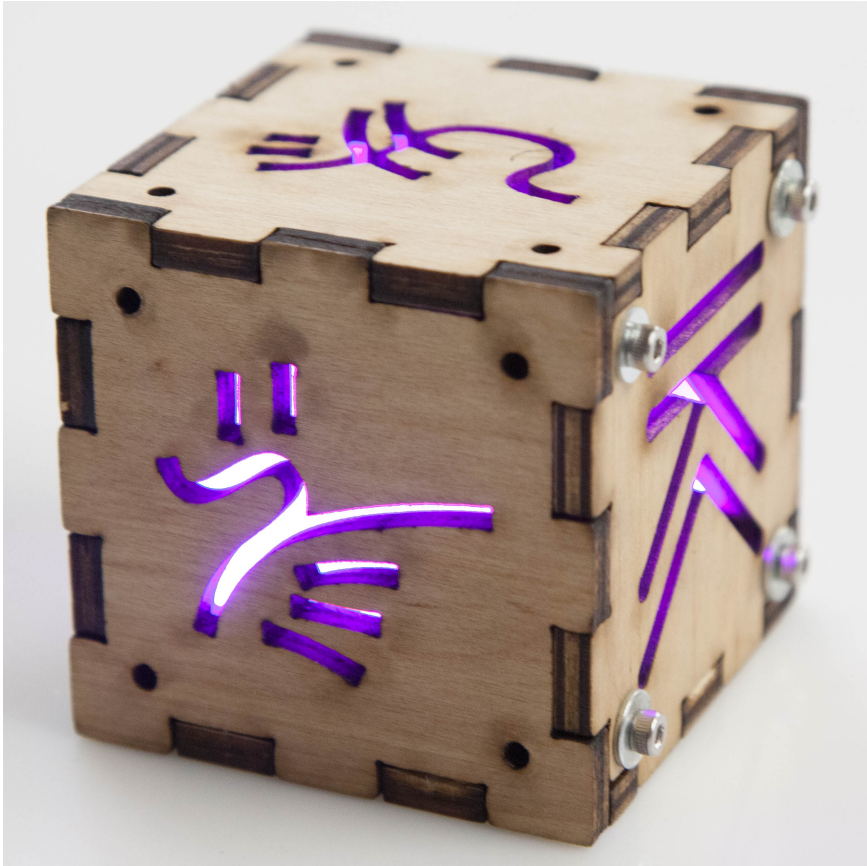

Figure 4.6: First wooden prototype of Transcode case. Note the hardware sitting above the face of the sides instead of flush. Photo credit: Mel McNiece

Figure 4.7: Final wooden form for *Transcode* control interface. Note the hardware countersunk and flush with cube faces as well as stained finish.

Chapter 5

Conclusions

The previous chapters have explored the cross-discipline research which culminated in two contributions to these disciplines. First, *dreamIO*, a tool for use in the design and creation of interactive installation art–acting as a contribution to both the fields of computer science and visual art. And, second, a standalone interactive installation, *Transcode*, which demonstrates the flexibility and capabilities of the tool, the underlying design philosophy–referenced as interactive 'application'–and acts as a new and authentic contribution to the field of visual art. I have shown how this installation was made possible not only through the *dreamIO* platform but also through an understanding of digital signals and information made possible with the study of computer science in combination with visual art.

Through this research I have shown that by leveraging several existing mechanisms–TUI, WSN, and network based API–into a hardware and software system, complex interactive works can be produced with very little overhead, especially in contrast with the pitfalls of existing tools. Furthermore, through a functional analysis of the system–focusing on real-world usage statistics–I have shown the capabilities and limitations of the system. Illustrating the places with room for improvement and the fundamental value of the system. Furthermore, borrowing from computer science, the concept of 'application' was defined in terms of interactive art. This concept was explored as a design philosophy enabled by the *dreamIO* system–through the WiFi based API–and the value of this methodology was examined within the context of the design and implementation of interactive art installation.

By utilizing this framework as the core building block for an interactive installation–*Transcode*–I showed the capabilities of the *dreamIO* system as a fully functional tool. While this installation is an excellent demonstration of the system's capabilities it also

functions as a new contribution in the field of visual art. By discussing the details of designing the piece and the motivations behind the particular interactive elements, as well as the physical aesthetic properties, I showed the malleability of the *dreamIO* framework as well as principals of interactive design related to the motivations of the piece. Specifically rendered as an experience of exploration and play for users interacting with the piece. each domain, Relational Aesthetics and Cybernetics, I have shown how this piece, and digital art more generally, can be observed as having a form emergent from the movement of information through the relationships enabled by interactivity and that the physical properties of the piece, while contributing to this information flow, are secondary or subservient to the amorphic nature of digital media.

Fundamentally, these two works–the framework and the installation–contribute greatly to each other and reflect the cross-disciplinary nature of my studies. And, while each piece can be observed independently, through the discussion of both pieces I have shown the relationship between these two domains: computer science acting as a mechanism for design and implementation of complex tools for novel tasks–and as a mode of thought unhinged from physicality–and visual art as a primary motivator for the creation of tools and processes–the application for the Science. At first glance these two domains seem to have little in common. However, at a fundamental philosophical level, each domain seeks to explore new ideas and modes of thought. In essence, computer science provides the tools for creation and problem solving and visual art provides motivation and a language through which to communicate. All the while, both domains share a cognitive space through which to explore and develop ideas.

Appendix A

Additional Information

A.1 dreamIO Properties

Table A.1: dreamIO properties

Processor	32-bit Tensilica processor at 80Mhz[17]
Memory	1Mb
Inertial Measurement Unit	InvenSense MPU-6050[22]
LEDs	12 x WorldSemi WS2812B[55]
Battery Charger	Microchip MCP73831[34]
Voltage Regulator	TPS61200[45]
Size (without battery)	60.0mm x 60.0mm x 2.0mm
Size (with battery)	60mm x 60mm x 7.5mm
Volume (without battery)	$7.2cm^3$
Volume (with battery)	$27cm^3$
Weight (without battery)	16.40g
Weight (with battery)	33.16g

A.2 Comparable Wireless Microcontrollers

Redbear WiFi Mini

The Redbear Wifi Mini[40] features a dual core configuration consisting of a Texas Instruments ARM Cortex-M4 processor for applications (user programming) and a dedicated ARM core for WiFi processing. The user programmable core runs at a clock speed of 80 MHz and has 256KB RAM with 1MB flash memory. The hardware measures 48 x 24 x 5mm, has 27 GPIO pins, and an integrated voltage regulator.

Particle Photon

The Particle Photon[37] features a single core Broadcom STM32F205 ARM Cortex M3 running at 120Mhz with integrated WiFi management. The board contains 128KB RAM and 1MB of flash memory, measure 36.58 x 20.32 x 4.32mm, and has 18 GPIO pins.

Sparkfun Thing

The Sparkfun Thing[46] contains an ESP8266 chip configured at 80mhz and is compatible with the *dreamIO* firmware. No information about the RAM or flash memory could be found. The dimensions of the device are not specified. The board is configured with an onboard voltage regulator, 10 GPIO pins, and a single cell Lipo battery charger.

Adafruit Huzzah

The Adafruit Huzzah[21] contains an ESP8266 chip configured at 80mhz and is compatible with the *dreamIO* firmware. The dimensions and memory of the board could not be found. The board has 9 GPIO pins and an onboard voltage regulator.

A.3 API Endpoints

A.3.1 Inputs

The API endpoints for sending data to a *dreamIO* board.

/leds

This endpoint accepts two different message formats. The first format accepts four 'int' values corresponding to:

$$LEDindex\ Red\ Green\ Blue$$

The second format accepts three 'int' values corresponding to:

$$Red\ Green\ Blue$$

The first message format allows users to control the Red, Green, and Blue level for a specific LED (defined by the LEDindex). The second format sets all of the onboard LED colour values to those specified in the message.

/update

This endpoint accepts an 'int'. It allows users to change the update interval (in miliseconds) of the *dreamIO* board. The update interval defines the length of time between transmission of OSC data (output from board). Currently when the board boots, this value defaults to 25ms.

/reset

This endpoint accepts any data type. Sending any message to this endpoint will hard reset the microcontroller on the board.

A.3.2 Outputs

The endpoints for receiving data from a *dreamIO* board.

/time

This endpoint contains a single 'int' expressing the length of time (in ms) that the board has been active.

/ypr

The endpoint contains three 'float' values expressing the yaw, pitch, and roll, respectively, of the board. The value of these data points is in the range of [0, 1].

/batt

This endpoint contains a single 'float' value expressing the measured value of the battery voltage. The value of this data is scaled to the range of [0, 1]

/side

This endpoint contains a single 'int' value expressing an estimate of which side of the board is facing down. The mapping is shown in table A.2.

Table A.2: dreamIO Orientation Mapping

Value	Side
-1	Unresolved
0	Bottom
1	Top
2	Back
3	Front
4	Left
5	Right

A.4 Voltage Measurements

Equation for voltage divider:

$$V_{out} = V_{in}\frac{r_2}{r_1 + r_2} \tag{A.1}$$

The voltage divider on the PCB is designed with:

$$R_1 = 2Mohm \tag{A.2}$$

$$R_2 = 330Kohm \tag{A.3}$$

Therefore:

$$V_{out} = V_{in}0.142 \tag{A.4}$$

This yields in a measurable voltage range from 0.368V to 0.595V.

The ADC of the ESP8266 is 10-bit and by dropping the two least significant bits the signal noise is reduced. These remaining 8 bits of data yield a resolution of 256 steps per volt.

$$ADCMeasurement = ADCResolution \times InputVoltage \tag{A.5}$$

Using equation A.5, the decimal value of the input voltages measured by the ADC range between 94 and 152.

Using these as minimum and maximum, the value of the ADC is scaled with equation A.6 in order to produce a output in the range [0, 1] where x is the value measured by the ADC.

$$y = \frac{x - min}{max - min} \tag{A.6}$$

A.5 Artistic Works by Steven A. Bjornson

Below is a presentation of a number of pieces I created during the term of the masters' degree:

Memory (test)

Memory (test) is a sculpture consisting of a laptop computer running specially designed software. The software takes the contents of the computer's random access memory (RAM) and converts this data into a live visualization on the laptop's screen. Because the computer's RAM stores information about the processes of the computer–used by the operating system and applications running on the machine–the visualization is a constantly changing stream of pixels. While sometimes the memory's content contains graphical images which are displayed coherently on screen, much of the data is incomprehensible. In essence, the sculpture is the revealing of the internal 'thoughts' of the computer with the visual form paralleling the complexity of human cognition and the difficult and often frustrating task of trying to converting thoughts into words.

A sample video of this work can be found at: `https://vimeo.com/sabjorn/memorytest`

Roomtone

Roomtone is an interactive sounds installation which consists of four loudspeakers places in the corners of a room and a microphone placed in the center. The microphone signal is delayed independently by different lengths of time–tuned based on the characteristics of the room–and sent to each speaker. Because the speakers and the microphone are co-located, sounds produced in the space become locked in the feedback path and will continuously be re-broadcast. This feedback continues until an end condition–the audio level reaching above a set threshold–is reached in which case the system is cleared and the process restarted (ad infinitum). The delays of the microphone signal modulates the feedback rate between the microphone and each

speaker and allows an extended time before the end condition is reached. By this time the space is filled with low harmonic tones which are constructed out of the input sounds and tuned with the delay lengths of the system.

This piece explores the nature of feedback in an audio system made possible through the 'slowing' down of the feedback rate normally induced by a speaker-microphone relationship. Furthermore, by having the feedback loop exposed to the real world, and allowing external sounds–people and noises–to manipulate the process, the piece explores the interactivity made possible through a simple feedback system.

Documentation of this piece can be found at: `https://vimeo.com/album/3265316`

Colours

Colours is a three part experimental film consisting of three separate visual scenes augmented with synthesized audio. The audio tracks for each scene were automatically generated using custom software which analyzed each video frame and created sounds based on the ratio of colours contained within the frame.

The purpose of this piece is to explore the relationship between signals in the digital domain through creating a process to map the signals across different modalities.

All three parts of this film can be found at: `https://vimeo.com/album/3564488`

Iterate 1

Iterate 1 is an experimental animation resultant from software designed to model the collage styles of Pol Bury. Originally, the software was created to automatically generate single images based on modeling the rotational manipulations in the Bury works. Eventually, I realized that large numbers of these individual images could be strung together to make an animation showing the range and effect of these manipulations over time.

This piece is an exploration of the processes behind the creation of time-based art and the inspiration which is generated through the development of a process. Furthermore, it is an attempt to connect with an artist from the past whose work is at home in modern digital processes.

This film can be found at: https://vimeo.com/sabjorn/iterate1

A.6 Software Libraries

Below are the software libraries used to create the *dreamIO* firmware.

Adafruit NeoPixel

Version 1.0.6 - available at https://github.com/adafruit/Adafruit_NeoPixel

ESP8266 Arduino

Version 2.0.0 - available at https://github.com/esp8266/Arduino

i2cdevlib

Version Unknown - available at https://github.com/jrowberg/i2cdevlib

OSC

Version 1.3.3 - available at https://github.com/CNMAT/OSC

The complete code and circuit diagram for *dreamIO* can be found at https://github.com/sabjorn/dreamIO/.

A.7 Code

A.7.1 Python Data Record

Below is the Python script used to record OSC data from the *dreamIO* for all experiments in section 3.5.

The complete code and circuit diagram for *dreamIO* can be found at https://github.com/sabjorn/dreamIO/.

```
1 import socket
2 import argparse
3 import copy
```

```
4  from time import strftime
5  import csv
6  import OSC
7
8  UDP_IP = '0.0.0.0'  # binds to all IPs.
9  UDP_PORT = 9999  # initial UDP port
10 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
11 sock.bind((UDP_IP, UDP_PORT))
12
13 if __name__ == "__main__":
14
15     parser = argparse.ArgumentParser()
16     parser.add_argument("endpoint", type=str,
17                         help='the dreamIO endpoint which is
                               being recorded')
18     parser.add_argument("--update", "--u", type=int, default
           =25,
19                         help='change the broadcast update
                               interval on dreamIO side')
20     parser.add_argument("--red", "--r", type=int, default=0,
21                         help='set the red value for all leds'
                               )
22     parser.add_argument("--green", "--g", type=int, default
           =0,
23                         help='set the green value for all
                               leds')
24     parser.add_argument("--blue", "--b", type=int, default=0,
25                         help='set the blue value for all leds
                               ')
26     parser.add_argument("--port", "--p", type=int, default
           =9999,
27                         help="the UDP recieve port to listen
                               on")
28     args = parser.parse_args()
29
```

```
30      endpoint = '/{0}'.format(args.endpoint)
31      receive_address = '', args.port  # blank = 0.0.0.0, which
            binds to all?
32
33      c = OSC.OSCClient()
34      c.connect(('esp8266-{0}.local'.format(endpoint[1:]),
            8888))
35      oscmsg = OSC.OSCMessage()
36      oscmsg.setAddress("/leds")
37      oscmsg.append(args.red)
38      oscmsg.append(args.green)
39      oscmsg.append(args.blue)
40      c.send(oscmsg)
41      oscmsg.clear()
42
43      # send
44      if(args.update is not 25):
45          print("sending new update rate")
46          oscmsg.setAddress("/update")
47          oscmsg.append(args.update)
48          c.send(oscmsg)
49
50      c.close()
51      oscmsg.clear()
52
53      # define a message-handle function for the server to call
            .
54      vals = {'time': -1, 'y': -1, 'p': -1, 'r': -1,
55              'batt': -1, 'side': -1, 'packet': -1}
56      out = []
57      buff = 256  # number of vals to collect before store
58      old_time = 0
59
60      # write header
61      today = strftime("%d-%m-%y-%s")
```

```
62      output_file = open('datarecord_{0}_{1}_{2}_r{3}g{4}b{5}.
            csv'
63                         .format(today, endpoint[1:-1], args.
                               update, args.red, args.green, args.
                               blue), 'wb')
64      dict_writer = csv.DictWriter(output_file, vals.keys())
65      dict_writer.writeheader()
66
67      try:
68          while 1:
69              data, addr = sock.recvfrom(2048)
70              data = OSC.decodeOSC(data)[2:]  # get rid of
                    header
71
72              vals['time'] = data[0][2]
73              vals['y'] = data[1][2]
74              vals['p'] = data[1][3]
75              vals['r'] = data[1][4]
76              vals['batt'] = data[2][2]
77              vals['side'] = data[3][2]
78              vals['packet'] = data[4][2]
79
80              print('recieved\tmess: {0}\ttime diff:{1}'.format
                    (
81                   len(out), (vals['time']-old_time)))
82              old_time = vals['time']
83
84              out.append(copy.copy(vals))
85              vals = dict.fromkeys(vals, -1)  # clear dict
86
87              if(len(out) > buff):
88                  dict_writer.writerows(out)
89                  out = []
90                  print('recorded')
91
```

```
92      except KeyboardInterrupt:
93          print("\nSaving File.")
94          if len(out) > 0:
95              dict_writer.writerows(out)
96          output_file.close()
```

A.8 Recorded Data

Tables A.3, A.4, A.5, and A.6 show the current draw of the development board. The current measurements were recorded at 3.7V with an Agilent E3631A[2] laboratory power supply with 0.001A current measurement precision. The update rate for all experiments was set to 25ms. These values are discussed in section 3.5.

Table A.3: dreamIO LED Current Draw, Red

Brightness	Current Draw (A)
10	.104
20	.108
30	.115
40	.123
50	.131
60	.140
70	.148
80	.155
90	.165
100	.172
110	.180
120	.188
130	.197
140	.205
150	.213
160	.221
170	.229
180	.237
190	.246
200	.253
210	.261
220	.270
230	.278
240	.286
250	.294
255	.298

Table A.4: dreamIO LED Current Draw, Green

Brightness	Current Draw (A)
10	.103
20	.108
30	.115
40	.123
50	.130
60	.139
70	.147
80	.155
90	.163
100	.170
110	.179
120	.187
130	.195
140	.202
150	.210
160	.218
170	.226
180	.234
190	.242
200	.250
210	.258
220	.266
230	.275
240	.283
250	.290
255	.295

Table A.5: dreamIO LED Current Draw, Blue

Brightness	Current Draw (A)
10	.103
20	.107
30	.114
40	.121
50	.128
60	.136
70	.144
80	.150
90	.158
100	.166
110	.174
120	.180
130	.188
140	.195
150	.203
160	.210
170	.218
180	.226
190	.233
200	.241
210	.249
220	.256
230	.264
240	.272
250	.280
255	.284

Table A.6: dreamIO LED Current Draw, All

Brightness	Current Draw (A)
0	.101
25	.130
50	.186
75	.243
100	.302
125	.360
150	.415
175	.464
200	.503
225	.546
250	.587
255	.594

Bibliography

[1] Ableton live. `https://www.ableton.com/en/`. Accessed: 2016-07-11.

[2] Agilent Technologies. *Agilent E3631A Triple Output DC Power Supply User's Guide*, 04 2014. Eighth Edition.

[3] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[4] Arduino: Open source microcontroller. `arduino.cc`. Accessed: 2016-06-13.

[5] Arduino 101. `https://www.arduino.cc/en/Main/ArduinoBoard101`. Accessed: 2016-07-11.

[6] Arduino history. `http://arduinohistory.github.io/`. Accessed: 2016-08-03.

[7] Tim Berners-Lee, Roy Fielding, and Henrik Frystyk. Hypertext transfer protocol–http/1.0. Technical report, 1996.

[8] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[9] Bluetooth SIG Proprietary. *Specifications of the Bluetooth System: Master Table of Contents and Compliance Requirements*, 12 2014. Rev. 4.2.

[10] Nicolas Bourriaud, Simon Pleasance, Fronza Woods, and Mathieu Copeland. *Relational aesthetics*. Les presses du reel Dijon, 2002.

[11] Kerry Brougher and Judith Zilczer. *Visual music: synaesthesia in art and music since 1900*. Museum of Contemporary Art, 2005.

[12] Paul Brown, Charles Gere, Nick Lambert, and Catherine Mason. *White Heat Cold Logic: Early British Computer Art 1960-1980*. Mit Press, 2009.

[13] Arthur C. Clarke. Hazards of prophecy: The failure of imagination. In *Profiles of the Future: An Enquiry into the Limits of the Possible*. Harper and Row, New York, second revised edition edition, 1973.

[14] Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation, Koninklijke Philips Electronics N.V. *Universal Serial Bus Specification*, 4 2000. Rev. 2.0.

[15] Steven C Dillon and Andrew R Brown. The educational affordances of generative media in arts education. *INTED2010 Proceedings CD*, pages 005311–005320, 2010.

[16] DLink. *D-Link DIR-625 User Manual*, 2006. Rev. 1.0.

[17] Espressif Systems. *ESP8266EX Datasheet*, 6 2015. Rev. 4.3.

[18] Firmata protocol. `https://github.com/firmata/protocol`. Accessed: 2016-07-11.

[19] Jayme Earl Hero. *Wireless sensor networks for interaction in virtual environments*. PhD thesis, Iowa State University, 2004.

[20] S Hsu and JD Tygar. Wireless sensor networks: a building block for mass creativity and learning. In *Proceedings of the ACM Creativity & Cognition, Understanding the Creative Conversation Workshop*, 2009.

[21] Overview - adafruit huzzah esp8266 breakout - adafruit learning system. `https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/overview`. Accessed: 2016-10-05.

[22] InvenSense. *MPU-6000 and MPU-6050 Product Specification*, 8 2013. Rev. 3.4.

[23] Hiroshi Ishii. The tangible user interface and its evolution. *Communications of the ACM*, 51(6):32–36, 2008.

[24] Frontpage - raspbian. `https://www.raspbian.org/`. Accessed: 2016-09-21.

[25] Carolyn L Kane. *Chromatic Algorithms: Synthetic Color, Computer Art, and Aesthetics after Code*. University of Chicago Press, 2014.

[26] Kicad eda. `http://kicad-pcb.org/`. Accessed: 2016-08-06.

[27] Kinect - windows app development. https://developer.microsoft.com/en-us/windows/kinect. Accessed: 2016-10-20.

[28] Nick Kruge and Ge Wang. Madpad: A crowdsourcing system for audiovisual sampling. In Alexander Refsum Jensenius, Anders Tveit, Rolf Inge Goddessøy, and Dan Overholt, editors, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 185–190, Oslo, Norway, 2011.

[29] Ben F Laposky. Oscillons: electronic abstractions. *Leonardo*, pages 345–354, 1969.

[30] Make a box. http://makeabox.io/. Accessed: 2016-06-22.

[31] Max for live. https://www.ableton.com/en/live/max-for-live/. Accessed: 2016-07-11.

[32] Max is a visual programming language for media. cycling 74. https://cycling74.com/products/max/. Accessed: 2016-10-26.

[33] Multicast dns. http://www.multicastdns.org/. Accessed: 2016-08-14.

[34] Microchip. *MCP73831/2 Miniature Single-Cell, Fully Integrated Li-Ion, Li-Polymer Charge Management Controllers Datasheet*, 9 2008. Rev. E.

[35] Henry Newton-Dunn, Hiroaki Nakano, and James Gibson. Block jam: A tangible interface for interactive music. In *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*, NIME '03, pages 170–177, Singapore, Singapore, 2003. National University of Singapore.

[36] Jieun Oh and Ge Wang. Audience Participation Techniques Based on Social Mobile Computing. In *Proceedings of the 2011 International Computer Music Conference*, Huddersfield, UK, 2011. University of Huddersfield.

[37] Particle datasheets documentation photon datasheet. https://docs.particle.io/datasheets/photon-datasheet/. Accessed: 2016-10-05.

[38] Raspberry pi 3 model b - raspberry pi. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. Accessed: 2016-09-21.

[39] Python Software Foundation. *The Python Language Reference*, 09 2016. Version 2.7.12.

[40] Redbear. `http://redbearlab.com/`. Accessed: 2016-10-05.

[41] Features - resolume vj software. `https://resolume.com/software`. Accessed: 2016-10-26.

[42] Charles Roberts, Angus Forbes, and Tobias Höllerer. Enabling multimodal mobile interfaces for interactive musical performance. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, pages 102–105, 2013.

[43] Bert Schiettecatte and Jean Vanderdonckt. Audiocubes: a distributed cube tangible interface based on interaction range for sound design. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 3–10. ACM, 2008.

[44] W Andrew Schloss. Using contemporary technology in live performance: The dilemma of the performer. *Journal of New Music Research*, 32(3):239–242, 2003.

[45] Texas Instruments. *TPS6120x Low Input Voltage Synchronous Boost Converter With 1.3-A Switches Datasheet*, 3 2013. Rev. D.

[46] sparkfun esp8266 thing: A breakout and development board for the esp8266 wifi soc. `https://github.com/sparkfun/ESP8266_Thing`. Accessed: 2016-10-05.

[47] Bernie C Till, Manjinder Singh Benning, and Nigel Livingston. Wireless inertial sensor package (wisp). In *Proceedings of the 7th international conference on New interfaces for musical expression*, pages 403–404. ACM, 2007.

[48] Touchosc by hexler. `http://hexler.net/software/touchosc/`. Accessed: 2016-07-11.

[49] transcode - definition of transcode in english from the oxford dictionary. `http://www.oxforddictionaries.com/definition/english/transcode`. Accessed: 2016-09-08.

[50] Unicode 9.0. `http://unicode.org/charts/`. Accessed: 2016-06-22.

[51] Unicode, Inc. *The Unicode Standard, Linear A, Range: 106001077F*, 2016. Version 9.0.

[52] Norbert Wiener. *Cybernetics or Control and Communication in the Animal and the Machine*, volume 25. MIT press, 1961.

[53] Different wifi protocols and data rates¡. `http://www.intel.com/content/www/us/en/support/network-and-i-o/wireless-networking/000005725.html`. Accessed: 2016-09-25.

[54] Wireshark go deep. `https://www.wireshark.org/`. Accessed: 2016-09-25.

[55] WorldSemi. *WS2812B Intelligent Control LED Integrated Light Source Datasheet*.

[56] Matthew Wright, Adrian Freed, and Ali Momeni. Opensound control: State of the art 2003. In *Proceedings of the 2003 conference on New interfaces for musical expression*, pages 153–160. National University of Singapore, 2003.