Dissertation outline/paragraphs: use this until I have the up-to-date latex template.

# Title page

CYBERINET: INTEGRATED SEMI-MODULAR SENSORS FOR THE  
COMPUTER-AUGMENTED CLARINET  
A Dissertation  
Submitted to the Graduate Faculty of the  
Louisiana State University and  
Agricultural and Mechanical College  
in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy  
in  
The School of Music  
by  
Matthew A. Bardin  
B.M., Stetson University, 2017  
M.M., The Boston Conservatory at Berklee, 2019

# Acknowledgements

First, I would like to thank my family. Without their continued support, I would not have been able to make it into graduate school, let alone be finishing my dissertation project. Thank you from the bottom of my heart Mom, Dad, Michael, Melanie, and everyone else. [If I listed everyone, this would have to be its own chapter]

I would also like to thank my various professors and mentors that I have had the pleasure of working with over the past decade. From building a general approach to music and technology in my undergraduate studies, to finding my passion while finishing my Master’s degree in Boston, to the guidance I received on this project; the advice I have received from these people has helped me to not only build a strong knowledge base, but achieve the confidence to present my work and my art to the world. [name drop here?]

Lastly, I would like to thank my cat, Bean. Regardless of whether she was in the way; she always made a spot on my desk to join me while I was writing. When in doubt, I would always defer to her expert opinion on any matters related to eat-scratches, pets, and wet food. Thank you, Bean.

[picture of bean]

# Abstract

The goal of this project is to provide a method of computer-enhanced performance to the solo clarinetist with minimal interference. To their normal routine. The end goal if for a performer to be able to seamless switch between a traditional performance setting and an augmented one with a press of a button. Towards this goal, the Cyberinet is a hardware replacement for a portion of a clarinet containing a variety of sensors embedded within the unit. These sensors collect various real time data such as gyroscopic data of the performer and air flow within the instrument. This data is then transferred to a computer via Bluetooth connectivity in order to use the data in any number of potential electroacoustic performance settings. In addition to the base unit which is embedded within a 3D printed clarinet barrel, the Cyberinet was also designed to accept various expansion units through USB-C ports on the side of the device. This allows for the unit to become customizable based on various performance needs. The final portion of this project consists of a collection of Max objects designed from the ground up to utilize the Cyberinet’s data in musical contexts. These Max objects were utilized in the performance of three new compositions for the instrument on April 18th, 2023 in the Digital Media Center on LSU’s Baton Rouge campus.

# Introduction

# Chapter 1: AUGMENTED INSTRUMENT DESIGN AND PERFORMANCE: AN INTRODUCTION AND LITERATURE REVIEW

# Sensors used in the Cyberinet

At its core, the Cyberinet contains a small collection of sensors build into the hardware. These include: [list of sensor names and what they do]. In addition to these sensors, the unit contain two USB-C connectors on its side. [show picture] The first one is intended for the button board expansion, which gives the performer access to two buttons positioned on the thumb rest of their instrument. The functionality of these buttons can then be programmed using Max or another audio programming environment. The second USB-C port is designed to connect to a variety of other sensors, allowing for the performer to adjust their setup for whatever is needed for a particular performance. This is the origin of the phrase “semi-modular” in the project title. At the time of writing, one expansion sensor has been created and tested. This is the [figure something out, a mic that will respond to volume and transmit a bang.] Additional expansions have been planned and explored in more detail in the “Further Directions” section of this document.

ESP-32

MPU-6050

TP-4056-C

The TP-4056 is a standard chip utilized for the charging of lithium ion polymer batteries. This variant utilizes a USB type C connector over the micro-USB connector of the base model and allows for both the charging of the battery as well as the distribution of power from that battery. When charging, it is the USB-C port that is connected to the power supply. As seen in the hardware diagrams, no other USB-C port is directly connected to this chip.

SDP-31

Originally released in [year], this is the newest hardware component in the Bill of Materials. This unit detects differential airflow pressure between two points as well as temperature. Due to the SDP-3X line’s small form factor, the Sparkfun Qwiic connector break out board was selected for simple installation. Two small tubes are attached to the ports and used to measure the air pressure difference between the inside of the clarinet mouthpiece and the outside of the instrument.

Because of it’s higher accuracy, the SDP-31’s temperature sensor is utilized over the MPU-6050. Temperature is measured in degrees Celsius.

LiPo Battery

In order to power the entire system, a lithium ion polymer battery is included inside the hardware. In order to charge the unit, users can utilize the provided USB-C cable and power adapter to the port located on the bottom of the unit [show picture]. From an empty charge, the unit can take between one and three hours to fully charge. The exact times can vary depending on the power adapter and whether any additional accessories are connected.

The Battery included is rated for 1200 mAh as an improvement over the 320 mAh battery used for initial testing and prototyping. This size battery allows the Cyberinet to have as much run time as possible while still maintaining the size characteristics of the hardware electronics. The approximate run times are as follows:

Powered on, not connected: 4 hours.

Powered on, Bluetooth connected, no expansions added: 4-5 hours.

Powered on, Bluetooth connected, expansions added: 3 hours

It should be noted that the device can also run when connected directly to the wall socket power supply. This will also charge the battery.

When the battery capacity becomes critical, the device will stop transmitting data, however the LEDs may stay on due to residual amounts of electricity present in the system. To avoid accidental drop outs during a performance, it is recommended to fully charge the device prior to any performances. Then the battery reaches 20% charge, the red LED on the device will illuminate, and then flash as shown below:

15%: 1 flash

10%: 2 flashes

5%: 3 flashes

[have the device auto power off? Look into doing that at 2%] More information about how the Cyberinet responds to internal battery levels can be found in the software discussion portion of this document.

All these chips were selected for their ability to be dropped into the main board without an extensive need for the user to solder many components, and their use of standard 0.1 inch spacing. [fix this sentence]

# Optional Expansions

These attachments are optional and not needed for the Cyberinet main unit to be functional and are intended to only be connected when needed for a particular performance. All the expansions connect utilizing a standard USB-C connector. However, these units do not utilize the USB protocol, so both the main unit and expansions cannot be connected to a computer via these ports. Because the Cyberinet does not communicate to these using USB protocol, not all USB-C cables can be used for this, however the vast majority can. [make this true] While a set of USB-C cables that are functional is included with the full Cyberinet set of hardware, The main reason for utilizing this connector is for the end-user to be able to supply their own cables of various lengths depending on their needs.

Button Board

This board, while optional, is incredibly useful for a performer on stage. They can access the buttons using their right thumb when performing. This maneuver is easier when utilizing a neck strap, so the performer can place the buttons elsewhere if they like using a longer connector cable. Each button also contains a single, colored LED built into it to provide feedback to the performer. By default, the lights are only illuminated when the button is pressed. [investigate a way to change that perhaps?] The Cyberinet simply detects whether a button has been pressed and transmits that data as a Boolean value to the computer. Using a program such as Max, the user can have the buttons achieve functions from near limitless hypothetical list of options. For this project, the buttons were used to trigger microphone recording and buffer playback, however objects that take the button input and move between various presets, trigger DSP, and turn pages of a score have also been developed and included in the software bundle for this project.

# Software

Arduino code

Max Library

To utilize the data being collected by the Cyberinet’s hardware, I have created a small library of Max objects to help with the collection and usage of said data. While detailed below, it is important to mention that Max is a paid software. The software and hardware components are all open source, however in order to create your own patches, the end-user will need to download the software. If saving patches is desired, then they must purchase a software license. Potential freeware options are being considered and will be explored upon more in a later chapter, but at the time of writing, are not in active development.

The main core of the Max library consists of two main objects. These are the .receive and .omnirecieve objects respectively. Looking at the former of the two, This object is designed to receive serial data from the Cyberinet, but filters out all data except that from a given sensor. This is useful for scenarios when the full functionality of the Cyberinet is not needed. .omnireceive, as the name implies, receives data from the Cyberinet as well and outputs the data from each sensor to unique outputs. This allows for the user to easily access any of the incoming data at once, and is useful for prototyping which sensor control could be useful for ran effect, or in a performance scenario when multiple sensors are required.

[talk about connecting to the serial port and how to make that reliable]

While not necessary for the functionality of the Cyberinet, I have also created a small collection of Max objects for use with the unit. These objects were all used within the Compositions discussed in chapter [give it a number] All of these objects are designed from the ground up to receive data from the Cyberinet in order to control their functionality. These include:

* Reverb
  + A Schroder style reverb based on examples from CCRMA.
* Compressor
  + Based on the compressor objects created by Cycling74.
* Filter
* Simple delay
  + A simple delay line using the [tapin~] and [tapout~] objects.
* Multi delay
  + A delay line utilizing 4 different [tapout~] objects to create multiple delay lines. Each one can have its individual output level set for unique mixing capabilities.
* Feedback delay
  + Identical to the simple delay, but with a feedback option added.
* Vibrato delay
  + Functions similar to the Simple [or was it feedback?] delay object, but includes a basic implementation of FM synthesis to adjust the pitch of the signal to be delayed.

[tutorial and help patches]

[make it into a standalone program? Maybe if there is time, maybe as a future endeavor]

# Works written for the Cyberinet

In addition to creating the hardware and software for this dissertation, I have also created a handful of musical compositions that show off various features of the Cyberinet. These three compositions as listed below, show a gradient from simpler to more complex uses of the Cyberinet’s features.

Puzzle of a Park:

At its core, this work is the simplest of the three written especially for this project. *Puzzle of a Park* functions as a work written for a loop pedal, but without the pedal. The performer plays through the material from beginning to end, periodically pressing one of the two buttons on the button board accessory. Looking at the max patch we can see that one button will trigger the computer to record a microphone input into a buffer while the second triggers the synchronized playback of all recorded buffers.

[show patch snapshot]

This functionality allows for multiple recordings to be saved and layered, much like loop pedals such as the Boss RC line of pedals. [do I talk about multi tracking?]

When writing this work, I viewed the musical content as a quartet, with four unique voices rather than a single voice repeated several times. This allowed the musical content to flow and feel more organic once the final layer was added.

Once I had completed the short musical section, it was simply a matter of repeating the sequence of time signature and meter changes so that the score would form a repeating pattern. From there I had to decide exactly how to organize the sounds in time. Because I wanted this to be the simplest of the works in terms of performance and programming, I refrained from breaking down the recordings into smaller chunks and assembling them in Max. Instead, I choose the loop pedal approach and had the performer play through each voice before starting the playback and recording for the following voice.

When ordering the voices, I began with one of the middle voices as the opening to the solo part. When comparing the voices in the score, these helped to provide a large amount of the background and a steady, albeit syncopated, pulse to the music. Something that helps to give the following bass voice more context during the measures where it is simply holding a pedal tone during the second section. Looking at the large-scale musical form of the solo, I would describe it as ABAC since the third and first voice are often coordinated, providing harmonic content. I viewed them like a horn part in a Sousa march in this regard. When combined, the first three voices provide a complete backing track to the main melodic line. Like fitting together, the pieces of a puzzle, it is this final line that helps to give context to all the phrases we have heard so far. Down beats become upbeats, harmonic implications shift with the addition of new chord tones, and the texture is filled out to include the full range of the instrument.

Ethereal Presence

The second work created for this project begins to utilize the more complicated sensors present in the Cyberinet. These primarily include the gyroscope and airflow meter. Once received, the Max patch for the composition utilizes the values to control two different synthesizers and their parameters.

[chart showing the value and parameter mapping]

The first synthesizer outputs a single tone with the goal of harmonizing with the soloist. This is the Ethereal Presence as mentioned in the title. [add more when this synth is done]

Accompanied by this is a more textural synthesizer designed to help support and fill out the atmosphere of the composition. This synthesizer outputs noise before being run through an FFT, based filter created by Dr. Austin Franklin [ how much info on the project do I need?]. This filter takes the incoming noise and performs an FFT operation on it to determine the frequency content, then based on the values received from the Cyberinet, filters out certain frequency bins.

Raindrops on a Tin Roof