Imperial College London
Department of Electrical and Electronic Engineering

Final Year Project - Interim Report 2015

PAWS
Programmable And Wearable Sound

Project Specification

'The development and evaluation of a novel and unusual musical instrument to be constructed using a 3D printer.'

The aim of this project is to research the current market and design a new musical instrument that integrates the traditional definition of being able to generate sounds with modern studio production methods and technological trends such as sampling, synthesis, and motion capture, in an effort to increase functionality while maintaining musical expression.

The product itself will be comprised of a wearable hardware sensor board and a counterpart software interface to control the audio output of each board. The sensor boards will be wearable through custom-designed 3D printed housings so as to meet the brief. The instrument will be produced in repeating production cycles, as per the iterative waterfall model, with each cycle building upon either the functionality or the aesthetic design. The instrument will also be evaluated in terms of usability with musicians and in terms of marketability against competing ventures.

Project	A Novel & Unusual Musical Instrument
Name	Kartiksinh K. Gohil
CID	00692607
Supervisors	Prof Robert Spence, Dr Mark Witkowski
Second Marker	Dr Christos Papavassiliou

Contents

1 Market Research

This final year project has a short brief (shown in italics in the Project Specification section) and is open-ended, relying more on a creative musically-oriented approach rather than the usual best-fit engineering solution.

The definition of a 'musical instrument' has evolved drastically over the years, ranging from traditional acoustic instruments (*piano, violin*) to electrical (*guitars, keyboards*), electronic (*synthesisers, Theremin*), and even virtual instruments that exist purely as software models in audio production tools.

The current music market has been overtaken by new computer vision and wearable technologies, giving way to products such as the Mi.Mu[1] gloves and DrumPants[3]. The idea of capturing motion through worn sensors or distant cameras allows the user to integrate their bodies with electronic systems. Musicians can make sweeping gestures to produce and control sounds, much like was traditionally done with mechanical instruments. With the initial introduction of electronic instruments, musicians became confined to button-based keyboards and drum machines, and later to point-and-click software on computers.

New technologies have allowed the modern musician to let the motion of their body contribute to their overall sound but they still seem to be fairly restrictive. The Mi.Mu gloves, as shown in Figure 1, allow the user to generate music simply by moving their hands. They can emulate playing a drum kit with their hands and the gloves will send MIDI signals to a digital audio workstation, which will produce the correct sounds in response to the user's 'air-drumming'. The gloves can also be used to control functions on sound, such as altering amplitude, filtering and even adding effects like Reverb, all through a pre-determined hand motion. This technology, however, is incredibly expensive. Selling at nearly £5000[2], these gloves are not accessible to the average person. It also requires other software tools for it to interact with, meaning that the user is in fact confined to a particular physical workspace, be it a studio or a live stage.

Figure 1: Mi.Mu Gloves[2]

DrumPants is another product that utilises wearable sensors to control sounds. The instrument, shown in Figure 2, uses sensor strips that attach to your clothes and connect wirelessly to a central controller. The strips contain pressure sensors that, when hit, send MIDI data to the controller, which can be set by the user to play any virtual sound in response, such as a drum kit or a piano. This controller can output audio directly to a speaker or can send the raw MIDI data to a compatible computer program.

Figure 2: DrumPants Basic Kit[4]

Currently on pre-order from $129.99[4], this instrument allows the user to program each sensor to play any pre-recorded sample sound, and is also being marketed to remotely control video games. The functionality, however, is limited to contact via pressure, and slightly malforms the user's experience by again restricting them to a particular physical area at any given instance. Even though this physical workspace, or the area where the sensor is located, is movable between instances of use, it still limits the musician's ability to improvise and requires an inordinate amount of setup time before playing can commence.

A third product, not specifically designed for music but important nonetheless, is the Ring[5] by Logbar Inc. The Ring has the capability of controlling any web-linked interface with gestures through an Android or iOS App. It is designed for the user to assign gestures to specific features, from controlling music playout on a smartphone to opening a set of shower curtains (provided they have internet connectivity). The idea behind the Ring is that it allows a user to directly interact with the world around them through a portable wearable sensor with apparently no physical limitations. The concept of this device, in its functionality and freedom of use, would greatly enhance a musician's creativity if applied to the world of sound.

Figure 3: Ring by Logbar Inc. [5]

Based on these new products, I aim to produce a musical instrument that heads in the direction of user-programmable electronics that is portable and not restricted to a physical workspace in the way that most

other motion-based products (especially computer-vision related) are. The musical instrument should allow the user to define the manner in which they wish to produce sounds and should allow them to use the range of their entire bodies in the process without introducing any physical or mental limitations.

2 Product Concept

The musical instrument I will be building will be based around the concept of programmable, wearable sensors. Firstly, the instrument shall use an array of sensors, such as audio (microphone) or motion (accelerometer/gyroscope), to capture and control sound. These sensors should be wearable by the user, most likely through bespoke 3D printed housings so as to meet that particular aspect of the project brief. The ability to wear these sensors in any location will increase the adaptability of their use, as well as removing the restrictions of a fixed physical environment. These sensors should also be programmable by the user in order to complete any given task. The user should be able to perform functions such as recording their voice, emulating a real live drum kit, mapping a tempo, and even harmonising their singing with virtual instruments, all by programming the sensors in various ways. Figure 4 shows a user-level diagram of a sensor array [PAWS Board] sending information wirelessly to an interface on either a computer or a smartphone and consequently producing audio.

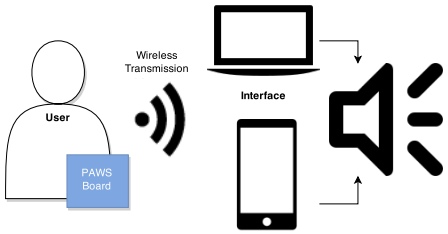


Figure 4: User Level Design

My instrument, entitled PAWS [Programmable And Wearable Sound], will in fact have both a hardware and software aspect to it. The hardware shall be a number of standardised sensor arrays that can be attached to any part of the body through custom 3D printed housings, which send their recorded data to a software interface, be it on a laptop or on a smartphone. This interface will allow the user to control the function of each PAWS Board and record the output audio to file, as shown in the concept design in Figure 5.

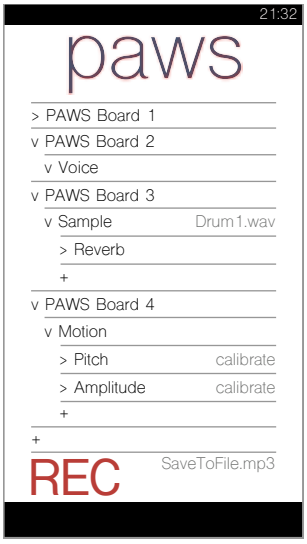


Figure 5: Concept Design for a smartphone-based Interface

The focus of this product is not on building perfect sensor arrays with minimum latency, or on developing a new signal processing technology, but rather on conglomerating the various existing ideas on the market into a single instrument that can be used by musicians in any way they like to accomplish any given task. The key features of this instrument, therefore, should be flexibility and simplicity of use.

Currently, the PAWS Board has three main functionalities. The first and simplest is to simply record vocals or any other sound that a PAWS Board may capture through its microphone. The interface should be able to obtain the input from the Board in question and save it to a file or play it back for the musician to listen to through the interface's built-in audio output. The second functionality the instrument should have is to be able to trigger sample sounds based on percussive motion. Figure 6 shows a finger with a PAWS Board attached tapping a rhythm on a random surface. If the Board has been programmed to trigger a sample such as a drum sound (as demonstrated in Figure 5), the user should be able to play a virtual drum kit through any given surface. Figure 6 also shows a hand with another PAWS Board attached at the wrist controlling select parameters of the output audio through motion gestures. This would be achieved by programming a range of motion to adjust a specific parameter such as Amplitude or Pitch.

Further designs for the PAWS Board include allowing a musician to set a particular tempo by tapping their feet (with a PAWS Board attached), which could also aim to quantise any other sounds that they produce, thus improving the quality of the generated sound. The sample trigger function could also be used to let a musician harmonise with themselves. For example, if they were singing into one PAWS Board and tapped another onto a surface, as if playing a piano, the instrument would trigger a piano sample at the same pitch as their vocal

melody. Multiple PAWS Boards for sample triggering could be used to allow a musician to play entire chords in harmony with their voice.

IMAGE MISSING -> finger tap + hand motion

Figure 6: Concept Sketch of Usage

3 Objectives & Specifications

This section outlines the objectives of the project and the specifications that the musical instrument is required to meet. These will also help to evaluate the implemented prototypes closer to the end of the project.

The main objective of this project is to build the PAWS musical instrument and evaluate it as a potential commercial product. The instrument itself will be built in iterative production cycles, or in stages of prototypes, where each prototype builds upon its predecessor in terms of either functionality or usability. The evaluation plan has been described in Section 7.2 as the method by which I will test how well the implemented prototypes meet these specifications.

The aim of the musical instrument is to allow anyone of any musical background to produce sounds in whichever manner they please and as quickly as possible without requiring any unnecessarily time-consuming setup of the instrument itself. Table 1 shows a list of the specifications the musical instrument is required to meet and the final prototypes will be evaluated against.

The PAWS Board
Must
Be able to capture audio
Be wearable through 3D printed attachments
Be easy to setup
Could
Be able to capture motion
The Interface
Must
Be easy to use
Be able to connect to PAWS Boards
Allow the user to control the function of every connected PAWS Board
Be able to process the audio signal from each PAWS Board
Be able to trigger sample sound files
Be able to output the generated audio to the core output device
Could
Save the output audio to file
Be able to map motion gestures to control parameters

Table 1: List of Product Specifications

4 Concept Design

This section outlines the design of the concept musical instrument. Section 4.1 tackles the design of the overall system including its various conceptualised functionalities, while Section 4.2 talks about the methods in which the system may be implemented.

4.1 System Design

Figure 7 shows the high-level system design of the instrument. The instrument consists of multiple hardware PAWS Boards, each with the ability to capture sound and motion data, connected to a central interface which processes the data and produces an audio output.

Figure 8 gives a more detailed view of the processes involved inside the PAWS Boards and the Interface. The Interface has the ability to receive data wirelessly from any number of PAWS Boards and the function of each can be selected by the user.

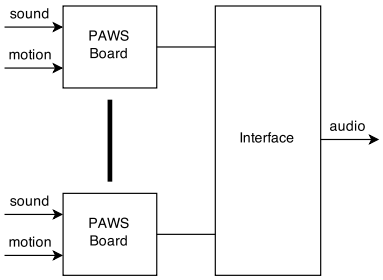


Figure 7: System Design - High Level

The Voice function simply routes the captured audio to the output. The Sample function allows the user to select a saved sample file and then processes the input data in order to trigger this sample. The Motion function allows the user to calibrate a particular gesture (such as sweeping a PAWS Board horizontally through the air) and assign it to change a particular modification parameter. For example, the selected parameter could be the global amplitude of the output audio, and the motion gesture would provide a continuous scale to change the amplitude level.

In this initial system design, the Motion function serves to control particular parameters of the global audio output, rather than a sound synthesis tool in itself. The Interface also allows the user to save the produced audio to a file.

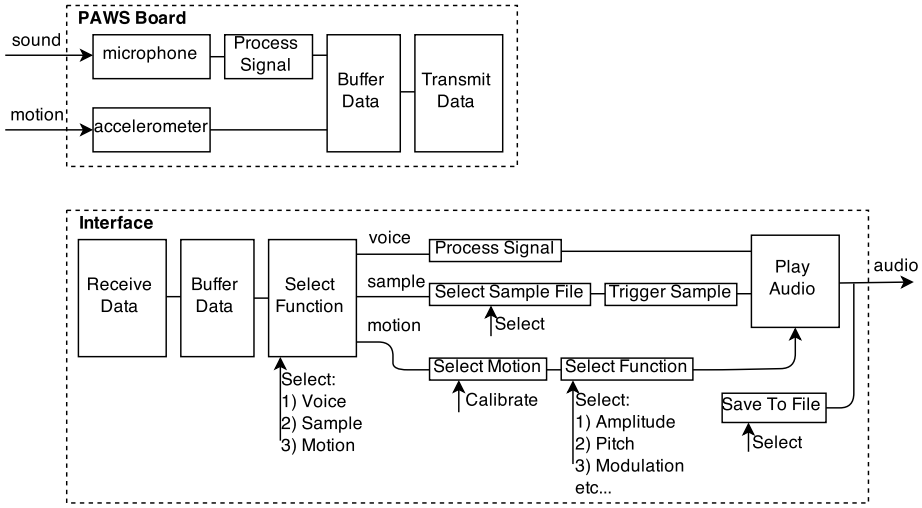


Figure 8: System Design - Low Level

4.2 Implementation Design

The PAWS Board hardware need to be light, portable and easily attachable to 3D printed wearable housings. This concept requires the production of printed circuit boards (PCBs) with components small enough to allow each Board to be placed anywhere on the body. During the initial production stages, various components and circuits will be tested on breadboard for functionality before being printed on a PCB. Initial implemented prototypes will also feature much bigger components that are much easier to test designs with, and therefore won't meet some of the required specifications of the overall product until much later in the production stage.

The Interface will initially be a computer application (specifically built for Mac OS X) as a starting point. It is required to provide a graphical user interface (GUI) to allow the user to control the signal processing chain. The easiest way to implement this program would be using the *Python* programming language. Python is an incredibly high-level language that is simple to write and can be used with a variety of signal processing and user-interface libraries. Libraries such as *Pydub*[14] and *PyAudio*[15] integrate the ability to manipulate audio in Python but did not specifically meet the requirements of the Interface software. The *PyO*[9] library is specifically designed for digital signal processing (DSP) in Python and will allow the Interface to be able to process audio signals in order to complete the user's tasks. The article *Python For Audio Signal Processing*[16] by Glover et al. demonstrates the use of other libraries for manipulating audio using Python, which may be useful when attempting to find the best solution to programming the Interface. Other languages such as *Objective-C* may also be explored for designing the graphical user interface, or *C* for a much lower-level (and therefore more efficient) implementation of signal manipulation techniques.

5 Initial Testing of Concept Design

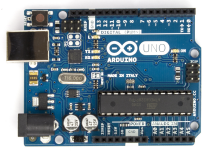
Before starting on the Implementation of the product, I tested several necessary elements such as using a microcontroller to read audio and output it to a connected terminal and I simulated the concept of sample triggering using Matlab.

5.1 Data Input and Output - Microcontroller

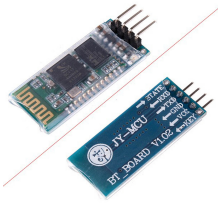
I tested the streaming of data in real-time by connecting a HC-06 (JY-MCU) Bluetooth Module to an Arduino Uno, both of which are shown in Figure 9. The Arduino Uno is a development board aimed at hobbyists built around an Atmel ATMEGA328P-PU microcontroller. The microcontroller can be programmed through Arduino's bespoke integrated development environment (IDE) that can be downloaded for free from Arduino's website[8] alongside documentation for the Uno board itself. The HC-06 Bluetooth Module can easily be attached to the serial outputs of the microcontroller through a simple circuit, shown in Appendix A.1.

The ATMEGA microcontroller can be programmed through Arduino's IDE to read analogue signals with its built in analogue-to-digital-converter (ADC) and output the read data to the HC-06 module via the Serial bus. In this sense, the microcontroller acts as a central digital processor that connects the output of the microphone circuit, shown in Appendix A, to the bluetooth module to be wirelessly transmitted to the instrument Interface.

I used a smartphone-based bluetooth terminal app to connect with the HC-06 module and ensure that it transmitted all the data sent to it by the microcontroller correctly. Once the ability of the microcontroller to read analogue signals and output them digitally to a bluetooth module were test sufficiently, I proceeded with the implementation of the first prototype of the instrument.



(a) Arduino Uno Development Board[7]



(b) HC-06 Bluetooth Module for Arduino[6]

Figure 9: Development Kit used for Prototyping

5.2 Data Input and Output - Interface

Certain elements of the Interface design were first tested using Matlab before implementing them in Python. I wrote Matlab scripts that would connect to the Serial Bus (via USB) and read the incoming data from the Arduino Uno. I used timed interrupts (in the way that the final implemented Interface would) to output the read data at a fixed sample rate.

After simulating the input and output of data on the Interface side, I moved onto implementing the software application using Python through an integrated development environment (IDE) called XCode.

5.3 Sample Triggering

Figure 10 shows the various processing stages of a simulation of triggering a pre-recorded drum sample from recorded audio. The raw recording was of a rhythm being tapped into the microphone of my smartphone. The recording then underwent very basic filtering and transient detection processes to produce spikes that were used to trigger the chosen drum sample. As we can see, the *microphone taps* were recognised correctly as triggers by the system and therefore produced a drum beat of the same rhythm, although slightly latent than the original recording. In this case, the raw recording was very clean, which may not be the case with the PAWS Board, and was processed offline whereas the PAWS Board's audio stream will be processed in real-time, and thus will be prone to more computational complications.

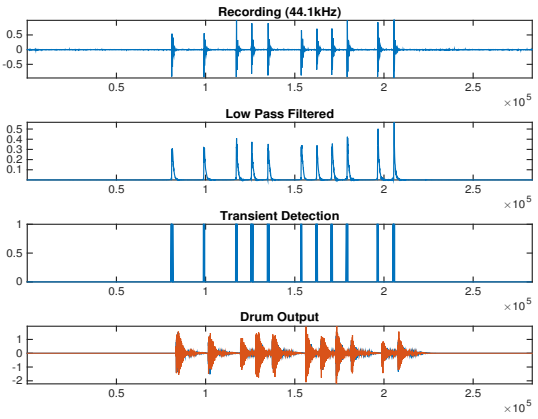


Figure 10: Simulation of Sample Triggering

6 Implementation

This section entails the implementation of the PAWS Board and counterpart Interface as a series of prototypes, with each version building upon the previous in terms of system design or aesthetics.

6.1 Prototype 0.1.01

photo of setup

Figure 11: Prototype 0.1.01 - Setup Photograph

Prototype 0.1.01 is the very first realisation of the PAWS musical instrument. Figure 11 shows the microphone circuit implemented on a breadboard and connected to the Arduino Uno development board. The breadboard also includes connections for a bluetooth module and a programmable gain amplifier (PGA) wired for testing but these have not been utilised for the current prototype. The circuit is comprised of a microphone connected to the Atmel ATMEGA328 microcontroller through a gain stage. The microcontroller is programmed to stream the audio data to the Serial bus, which is currently connected to the Interface on my laptop via a USB cable. The Interface was programmed in Python to read data from the Serial bus and output it to the laptop's core audio device. A library called *PyO*[9], written for implementing digital signal processing (DSP) functions, was used to generate two python scripts: one to play whatever audio the microphone captured, and another to trigger drum samples in response to the microphone being tapped with a finger.

This prototype is able to play the audio captured by the microphone, although it is fairly distorted. The prototype is also able to trigger drum sounds when the microphone is slapped. However, the interface is not yet programmed to filter the raw input and detect the transients in a clean manner and therefore the drum sounds have a slight latency and sometimes misfires.

6.1.1 Microphone Circuit

Figure 12 shows the microphone circuit used in Prototype 0.1.01. The first stage, connected as described in the datasheet[10], is fed into a TL072 op-amp with a gain of ten thousand to boost the audio signal to within the dynamic range of the microcontroller's ADC. This prototype uses the 5V power rails provided by the Arduino, and is therefore laced with digital noise. Thus, a 100μF electrolytic capacitor was placed across the rails in an attempt to reduce this noise.

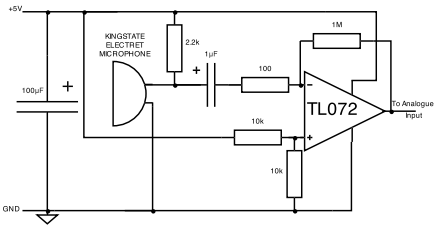


Figure 12: Prototype 0.1.01 - PAWS Board Circuit

6.1.2 Microcontroller Program

The microcontroller program uses a timed interrupt to read a value from its input analogue pin at a set frequency of 8kHz and this value is added to a global circular buffer. In the main program loop, the values in this circular buffer are printed to the Serial bus. The print to Serial cannot be handled inside the interrupt function as it takes much longer to process than the frequency of the function call. Therefore, the print to Serial process is handled in the main loop, however it can lead to inaccuracies due to the two functions writing to and reading from the circular buffer at different frequencies. A dynamically allocated queue would be preferable but the Arduino libraries *QueueList*[11] and *QueueArray*[12] seemed to be unable to cope with the speed at which the data was being read. Appendix B.1 shows a flowchart diagram of the microcontroller program.

6.1.3 Interface - Input and Output

Programming the Interface to read from the Serial bus is well-documented in the *pySerial*[13] library and was therefore very straightforward to implement. Each read sample was appended to python list, used as a buffer queue, as a *PyO* signal, ready for processing.

The first version of the Interface was developed to simply take the input signal buffer and output it to the core device, and thereafter clear the samples that had been played from the buffer. The *PyO* library uses an audio server that receives data samples asynchronously and outputs them at the user-defined sampling rate. Therefore, it was simple enough to program an infinite loop to constantly read in samples from the Serial bus and write them to the output server. Appendix B.2 shows a flowchart describing the processes involved in this Interface program.

6.1.4 Interface - Sample Triggering

Another version of the Interface was created where data samples were read in chunks from the Serial bus, compared against an intuitively-set threshold value and the generated on/off pulses were used to send the waveform of a drum sound to the output audio server. This method was done in the most basic manner in order to quickly demonstrate that it could indeed be done. The prototype was able to play drum sounds when the microphone was struck but there seemed to be a small delay between the two events. There were also many misfires, or drum sounds being played when they were not supposed to, due to the quickly implemented spike detection method. Appendix B.3 shows how this particular prototype implements the sample triggering function.

6.2 Further Prototypes

Table 2 shows a list of the implementable prototypes and their improved features.

Prototype 0.1.02 will include significant improvements over 0.1.01. On the hardware side, it will include an independent power supply (battery) for the microphone circuit in order to ensure it is not corrupted by the microcontroller's digital noise. The Atmel ATMEGA328 microcontroller will also feature a more robust method

of reading and writing the microphone signal such that we avoid the inaccuracies of the currently implemented circular buffer. The Interface will integrate the 'Voice' and 'Sample Triggering' functions into a single program and will perhaps even feature a basic GUI to allow the user to control the function of the PAWS Board. Once all of the initially specified system components have been implemented in some way, Prototypes 0.2.xx onwards will involve expanding the functionality of the instrument to give the user more creative control over the sound they wish to produce. Further prototypes will also explore marketable designs of the instrument, including more professional microcontrollers for smaller PAWS Board package designs, and Interface functions coded in lower-level programming languages such as *C* to optimise the signal processing chain.

Prototype	Features
0.1.01	Basic output of microphone and simple Sample Triggering function
0.1.02	Cleaner microphone signal, single GUI[1] for Interface with both Voice and Sample functions
0.1.03	PGA[2] in circuit for digital gain control
0.1.04	Bluetooth transmission
0.1.05	A fully featured GUI for user control
0.2.xx	All hardware on PCB[3] and 3D printed wearable housings
0.3.xx	Integration of SP[4] functions such as Gain, EQ[5], Pitch Detection & Correction
0.4.xx	Quantisation Function[6]
0.5.xx	Harmonisation Function[7]
0.6.xx	Integrate Accelerometer/Gyroscope onto PAWS Board and develop Motion feature on Interface

Table 2: List of Prototypes and their Features

[1] GUI: Graphical User Interface
[2] PGA: Programmable Gain Amplifier
[3] PCB: Printed Circuit Board
[4] SP: Signal Processing
[5] EQ: Equalisation
[6] Quantisation Function: User can tap a tempo to quantise all produced sample sounds
[7] Harmonisation Function: User can trigger virtual instrument samples to harmonise with their singing

7 Project Plan

This section briefly describes the progress plan of the Project including information on the iterative implementation of prototype designs and the evaluation of the overall success.

7.1 Implementation Plan

Table 3 shows the list of Prototypes from Table 2, Section 6.2, and their estimated completion dates, alongside the submission dates for all reports and the final presentation.

Prototype	Completion Date (2015)
0.1.01	20th January
Interim Report	2nd February
0.1.02	11th February
0.1.03	17th February
0.1.04	20th February
0.1.05	28th February
0.2.xx	18th March
0.3.xx	1st April
0.4.xx	17th April
0.5.xx	1st May
0.6.xx	15th May
Abstract	8th June
Final Report	17th June
Presentation	TBC

Table 3: Implementation Timetable of Deliverables

Prototype 0.1.05 should be functioning by the end of February 2015. This prototype should meet all of the basic specifications of the musical instrument. Subsequent prototypes will improve upon either the functionality, what the user can do with the instrument, or usability, how the user can interact with the instrument, and I have given myself approximately half a month to ensure each prototype is fully functional.

Detailed plans of implementation for each subsequent prototype will be drawn up as needed in order to ensure the successive completion of all the necessary design components. This production cycle where each iteration of prototypes builds upon the previous, thus ensuring that, no matter the progress, there will always be a version of the musical instrument that is functional across all of its basic requirements.

7.2 Evaluation Plan

The plan to evaluate the success of the musical instrument, and therefore of the project, mainly revolves around its ability to produce sound. There are certain specifications highlighted in Section 3 that must be met in order to enable my musical instrument to contend with the currently developing technologies.

Each fully built prototype will undergo a series of quantitative such as calculating the audio latency and distortion levels, as well as more qualitative criteria such as expected performance, user-friendliness, etc. These will help to identify the achieved improvements of each prototype and would also possibly help to reconsider the direction in which subsequent prototypes would have to be taken if certain criteria are not in fact met.

Shortly before the presentation and final report, when my prototypes are at a stage where they can be demonstrated fairly easily, I will conduct several field tests as part of my evaluation of the instrument as a potential commercial product. The musical instrument is aimed at allowing anyone, musically experienced or not, to generate their desired sounds quickly and easily without having to go through extensive setup processes. Therefore, I will take my final prototypes to the general public to gauge their interest and to further evaluate how well the product meets its user-based specifications. I will test my product with students and young

adults of ages 18-24 as they are able to adapt to new technologies fairly quickly and I will therefore be able to efficiently test my prototypes with as many as possible.

8 References

[1] Mi.Mu Gloves. *Mi.Mu: About.* [Online] Available from: http://mimugloves.bigcartel.com/mi-mu-bio [Accessed 24 Jan 2015].

[2] Mi.Mu Gloves. *Mi.Mu Collaborator Gloves.* [Online] Available from: http://mimugloves.bigcartel.com/product/collaborator-gloves [Accessed 25 Jan 2015].

[3] DrumPants Inc. *DrumPants.* [Online] Available from: http://www.drumpants.com/ [Accessed 24 Jan 2015].

[4] DrumPants Inc. *DrumPants.* [Online] Available from: https://legacy.trycelery.com/shop/drumpants [Accessed 25 Jan 2015].

[5] Logbar Inc. *Ring.* [Online] Available from: http://logbar.jp/ring/en [Accessed 25 Jan 2015].

[6] Ebay. *5V JY-MCU HC-06 V1.04 Bluetooth Transeiver RF Module Serial Port 51 AVR MCU ARM* [Online] http://www.ebay.com/itm/5V-JY-MCU-HC-06-V1-04-Bluetooth-Transeiver-RF-Module-Serial-Port-51-AVR-MCU-ARM-/261121350024 [Accessed 28 Jan 2015].

[7] Arduino. *Arduino Uno.* [Online] Available from: http://arduino.cc/en/main/arduinoBoardUno [Accessed 28 Jan 2015].

[8] Arduino. *Arduino.* [Online] Available from: http://arduino.cc [Accessed 28 Jan 2015].

[9] Olivier Bélanger, Ajax Sound Studio. *PyO 0.7.3 Documentation > API Documentation > Classes By Category.* [Online] Available from: http://ajaxsoundstudio.com/pyodoc/api/classes/index.html [Accessed 28 Jan 2015].

[10] Premier Farnell plc. *Specification for Approval - KINGSTATE, Electret Condenser Microphone, KEEG1542PBL-A.* [Online] Available from: http://www.farnell.com/datasheets/1780663.pdf [Accessed 12 Jan 2015].

[11] Efstathios Chatzikyriakidis. *QueueList Library for Arduino.* [Online] Available from: http://playground.arduino.cc/Code/QueueList [Accessed 17 Jan 2015].

[12] Efstathios Chatzikyriakidis. *QueueArray Library for Arduino.* [Online] Available from: http://playground.arduino.cc/Code/QueueArray [Accessed 17 Jan 2015].

[13] Chris Liechti, SourceForge. *Welcome to pySerial's documentation.* [Online] Available from: http://pyserial.sourceforge.net [Accessed 29 Jan 2015].

[14] Jiaaro. *Pydub by jiaaro.* [Online] Avaliable from: http://pydub.com [Accessed 1 Feb 2015].

[15] Hubert Pham. *PyAudio 0.2.8 documentation > PyAudio Documentation.* [Online] Available from: http://people.csail.mit.edu/hubert/pyaudio/docs/ [Accessed 1 Feb 2015].

[16] John Glover, Victor Lazzarini, Joseph Timoney. Python For Audio Signal Processing. The Sound and Digital Music Research Group. [Online] 2011. Available from: http://lac.linuxaudio.org/2011/papers/40.pdf [Accessed 1 Feb 2015].

[17] techbitar. *Cheap 2-Way Bluetooth Connection Between Arduino and PC.* [Online] Available from: http://www.instructables.com/id/Cheap-2-Way-Bluetooth-Connection-Between-Arduino-a/ [Accessed 1 Feb 2015].

A Circuit Diagrams

A.1 Circuit Diagram for HC-06 - Arduino Connection

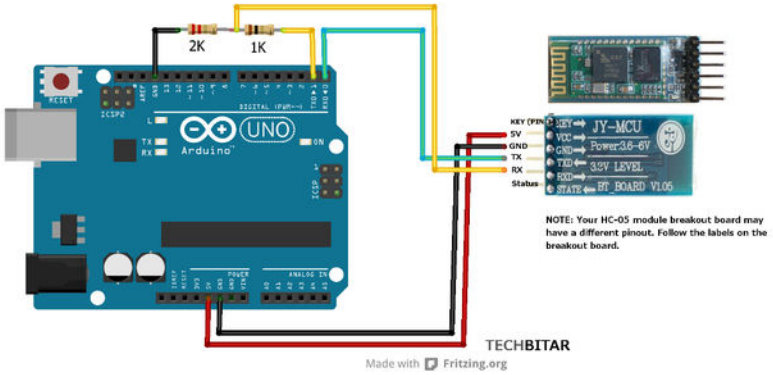
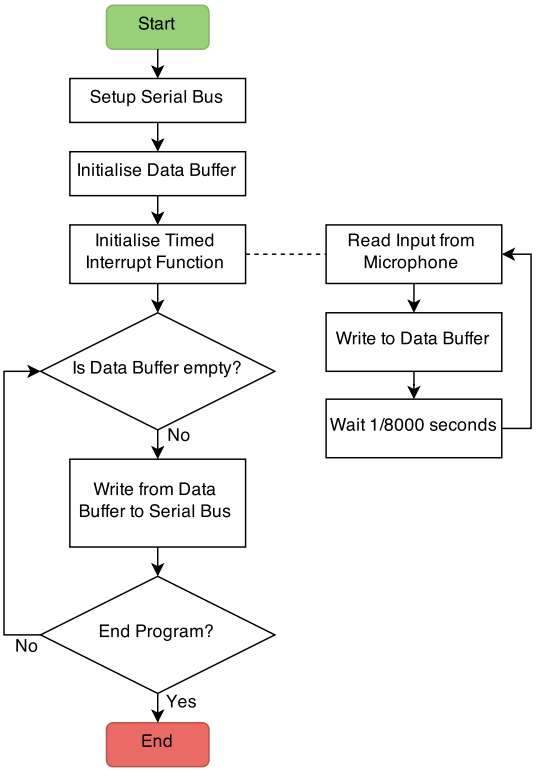


Figure 13: Image Available from Reference [17]

B Flowchart Diagrams

B.1 Flowchart for Microcontroller Program, Prototype 0.1.01



B.2 Flowchart for Interface Program - Voice, Prototype 0.1.01

B.3 Flowchart for Interface Program - Sample, Prototype 0.1.01