Shoes

A fully wireless, wearable foot controller for live performance

Contents

1. Project Overview

1.1 Introduction

Shoes is a fully wireless, wearable, entirely foot-controlled instrument/interface, housed in a pair of Adidas hi-top sneakers.
Shoes expands the expressive capabilities of the feet, allowing them to be used as the chief creators and/or augmenters of sound in interesting and engaging musical performances.
This is achieved by housing a number of analogue sensors in the shoes (details in 3.2 - Hardware), which detect changes to several parameters that the feet can control. In choosing these parameters, I wanted to ensure that there were a sufficient number to give a high level of expressivity, but not so many that *Shoes* was physically cluttered, and performance therewith overly complex.
With this in mind I decided to focus on:

- Force, at pressure points in the toe and heel. Used for both expressive volume control and button-esque functions.
- Flex, at a bend point on the top of the left shoe. Used for fine-grain expressivity such as vibrato, tremolo, and pitch bends.
- Proximity, between the two feet. Used chiefly for pitch control.

Another essential element of *Shoes* is its wirelessness. Using a pair of WiFi-capable microcontrollers (details in 3.2 - Hardware) I was able to make each shoe require no external wiring, either for data transfer or for power supply. This greatly reduces performer encumbrance and makes *Shoes* far more convenient and user-friendly as an instrument.
Shoes is still a work in progress, and I have not fully realised my vision for it yet. Nonetheless, the current prototype gives a good idea of my intentions, and is a strong base upon which to improve further. The following is a written documentation of that prototype.

1.2 Motivation

When it comes to traditional performative instruments, feet are generally used in only a very rudimentary way. When they are being used for sound *production*, the sound is usually simple and rhythmic - kick drums, stomp boxes etc. When the feet are being used in the creation of more complex sound, it is almost always only as an *augmenter* of sound - guitar pedals, piano pedals etc., and even then the use is still rudimentary; one point of control, on/off.
I believed that expanding the expressive capabilities of the feet and allowing them to move beyond their traditional performative role, through the development of a new

instrument/interface designed specifically for them, could open up an exciting new realm of performative possibilities.
First of all, *simultaneous* use of this new instrument alongside a traditional, hand-controlled instrument would allow a single performer a new degree of depth and complexity in their performance. I, a saxophonist, had experimented with this before, using market-available foot controllers, but was not happy with the results. I believed I could do better!
Second of all, this new instrument could make performative music more *inclusive*. Almost all traditional instruments require full, fine control of a working set of hands in order to perform with them. This means that those with upper-body disabilities are faced with a serious obstacle when hoping to engage in performative music. The instrument that I intended to build would require no use of the hands whatsoever, and would thus present no such obstacles.
It was with these motivations in mind that I set out to design and build *Shoes*.

2. Related Works

2.1 KMI Softstep

The KMI Softstep, pictured above, developed by Keith McMillen Instruments and released to market in 2011, is a non-wearable foot-controlled interface. It, too, seeks to expand the expressive capabilities of the feet by expanding the number of parameters they can control, though it takes a slightly different approach to *Shoes*. Each pad can detect pressure, allowing for simple button press motions as well as expressive fades. The pad can also detect the x/y positioning of the foot at the time the pad is pressed/held. This is achieved using a ‘smart-fabric’ housed just beneath the rubber outer-layer.
The Softstep was my first foray into foot-controlled interfaces. I bought one with the intention of implementing a live-harmoniser that would allow me to play two saxophone parts at once during my live performances as *Sheep, Dog & Wolf*. In my use of the Softstep, I found it limiting in a couple of ways:

- *Restricts movement*: When using the Softstep, much like a guitar pedal, the performer is bound to the part of the stage where the Softstep is, greatly restricting any movement as a part of the performance.
- *Discrete control only*: Because the Softstep is controlled chiefly by a set of pads, pitch control for my live harmoniser was limited to 10 discrete values, the pressure sensitivity not providing enough range to be viable as an alternative form of control.

The above limitations result in the KMI Softstep not feeling at all like an expressive instrument, but rather a glorified effects pedal. Perfect for many use cases, I am sure, but not exactly what I was looking for.
More information about the Softstep can be found here:
https://www.keithmcmillen.com/products/softstep/

2.2 Rhythm'n'Shoes

The Rhythm'n'Shoes, pictured above, was developed by Stefano Papetti, Marco Civolani, and Federico Fontana, and presented at NIME 2011. The Rhythm'n'Shoes is similar in its intended use to *Shoes*, though more specific - it is a wearable, semi-wireless, foot-controlled instrument, designed solely for percussive performance. Force Sensitive Resistors are placed at the toe and heel of each shoe, the data from which is interpreted by an offboard computer and used to generate percussive sounds, as well as generating haptic feedback at the point of impact.
The shoes are 'semi' wireless, in that data transfer from the Arduino, which gathers the data, to the offboard computer is wireless. However only a single Arduino is used, and it is housed in a backpack worn by the performer, with wires from the shoes to the backpack going up the performer's legs and back. Great lengths were gone to to ensure very low latency, to accurately imitate the feel of a percussive instrument.
While there is no documented use of the Rhythm'n'Shoes alongside a traditional instrument in simultaneous performance, I am sure it could be used as such. The developers also note that the Rhythm'n'Shoes is highly inclusive, allowing not only those with upper-body disabilities to engage with it, but even people suffering partial or

complete hearing loss, due to the haptic feedback. Documentation for the Rhythm'n'Shoes can be found here:
http://www.nime.org/proceedings/2011/nime2011_473.pdf

2.3 NIKE MUSIC SHOE

The NIKE MUSIC SHOE, pictured above, is a strange one. While it is an indeed an instrument/interface housed within a shoe, from what I can tell by watching the demo videos it is not, in fact, really intended to be used as a foot-controlled interface.
It was developed by W+K Tokyo and released in 2010, commissioned by Nike as part of an advertising campaign to showcase the extreme flexibility of their new model of shoe, the Nike Free Run+. As such, the entirety of the expression is derived from 3 flex-sensors placed in the sole of the shoe, and an accelerometer to roughly detect orientation. Performers twist and bend the shoes, connected to a computer via cable, to affect sound.
It is difficult to tell how much control the performers actually have over the instrument - it seems to me that, despite the expressive range that flex provides, the shoes are still mostly used for simply triggering samples when a sharp change is detected. Most of the control over what sounds will be played still appears to lie with the computer.
Documentation and demonstrations of the NIKE MUSIC SHOE can be found here:
http://wktokyo.jp/works/nike-music-shoe/

3. Technical Details

3.1 Parts List

The parts used in this interface (excluding the enclosure materials) are listed below.

Right Shoe:

- 2x Force Sensitive Resistor (0.5” sensing area)
- 1x IR Proximity Sensor (10-:150cm range)
- 1x Particle Photon Wi-Fi Capable Microcontroller
- 1x Particle Power Shield
- 1x 3.7V 100mAh Li-Po Battery
- 2x 10kΩ resistor (for FSRs)
- 1x toggle switch

Left Shoe

- 2x Force Sensitive Resistor (0.5” sensing area)
- 1x 2.2” Flex Sensor (Longer is also acceptable)
- 1x Particle Photon Wi-Fi Capable Microcontroller
- 1x Particle Power Shield
- 1x 3.7V 100mAh Li-Po Battery
- 2x 10kΩ resistor (for FSRs)
- 1x 47kΩ resistor (for Flex Sensor)
- 1x toggle switch

3.2 Hardware

Below is a list of the key hardware elements, and how they are used in *Shoes*. For details on how these components are integrated, see 4.2 - Component Housing.

Sensors:

- *Force Sensitive Resistors:*
  FSRs are attached the toe and heel of each shoe. These FSRs detect both impact events, through software interpretation of their data, and expressive pressure swells.
- *Flex Sensor:*
  A flex sensor is attached to the top of the left shoe. This is used to detect bending of the left foot.
- *IR Proximity Sensor:*
  An IR Proximity sensor is attached to inside heel of the right shoe. This is used to detect the distance between the two feet.

Microcontroller Bundle:

- *Particle Photon Microcontroller:*
  One of these WiFi capable microcontrollers is mounted directly onto the Power Shield (see below) and attached to the outer heel of each shoe. It gathers data

from the potentiometers, packages them, and sends them wirelessly to an offboard computer (see 3.3 - Firmware and Software).

- *Particle Photon Power Shield:*
  The power shield regulates power supply to the microcontroller from a LiPo battery and allows for USB recharging of said battery.
- *3.7V 100mAh LiPo Battery:*
  This very small battery is attached to back of the Power Shield, and connected to it via a toggle switch that controls power supply.

3.3 Firmware and Software

Firmware:

The firmware running on each microcontroller is fairly simple, and almost identical. The data from each potentiometer is read every 5ms from the analog inputs and stored in a variable - for the right shoe, the IR data is also smoothed by taking an average of the current and preceding readings. Each of these variables is then hard-coded into a string that fits OSC formatting guidelines, through the use of low-level bit-shifting operations. The OSC string is then transmitted via UDP to the offboard computer. The UDP transmission protocol was selected, as opposed to TCP/IP, due to its prioritisation of speed over accuracy. High responsivity of the instrument was essential, and with such a large number of values being sent, a packet being dropped every now and then was not going to be an issue. UDP also had the benefit of being natively supported by Max/MSP, as long as the string being received was OSC formatted, removing the need for any middleware.

Software:

The software being run on the receiving computer varies based on what application *Shoes* is being used for. There are several stages of data-processing, some of which are not necessary for some applications. All software is written in Max/MSP.

Stage One: Max Patch 1 - Receiving/Unpacking Data (essential):

The OSC formatted string for each shoe is received into a Max patch via the udpreceive object. The string is then unpacked into individual data points. The patch then outputs the raw analog data for each potentiometer.

This stage is ‘essential,’ and occurs no matter what application *Shoes* is being used for. If the user of *Shoes* wishes to work with the raw data directly, this is the last software module required.

Stage Two: Max Patch 2 - Interpreting Data

The unpacked data is received into a second Max patch, which interprets the data and outputs relevant MIDI information for each potentiometer (if outputting

to a MIDI instrument) or the treated raw data (for native sound generation). This information can then be output directly to external sound-generation software (DAW, virtual instrument etc.) or to the next Max patch for native sound generation. If the user of *Shoes* wishes to generate sound externally, this is the last software module required.

Stage Three (A): Max Patch 3 - Solo Instrument

The treated data is used to generate sound natively within Max, to be output through an audio output device. At this stage there is only a single performance style and sound type, although in future there will be a number of performance styles and sound types, which will be able to be selected by the user during performance.

For an in-depth description of how this part of the software functions now, and will function in future, see 5.1 - Solo Instrument.

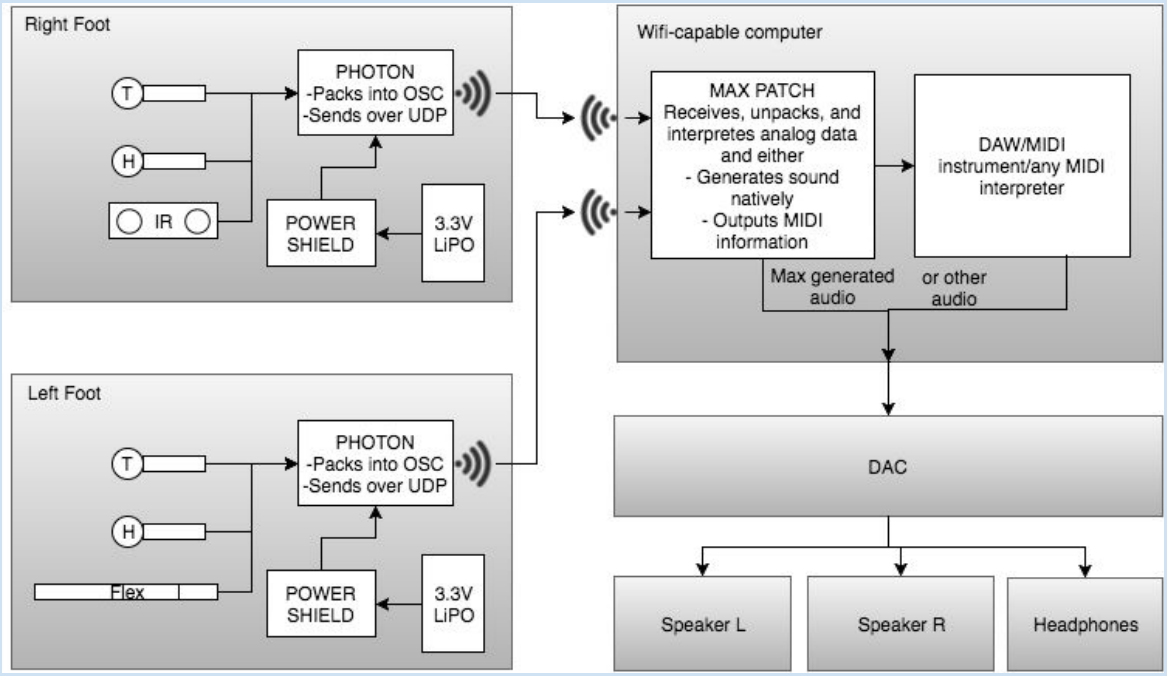
Stage Three (B): Max Patch 4 - Performance Augmentation

The treated data is used to alter and augment input audio, with the affected audio being output through an audio output device. The provided software is a live harmonisation patch, although there are many further possibilities.

For an in-depth description of how this part of the software functions, see 5.2 - Performance Augmentation

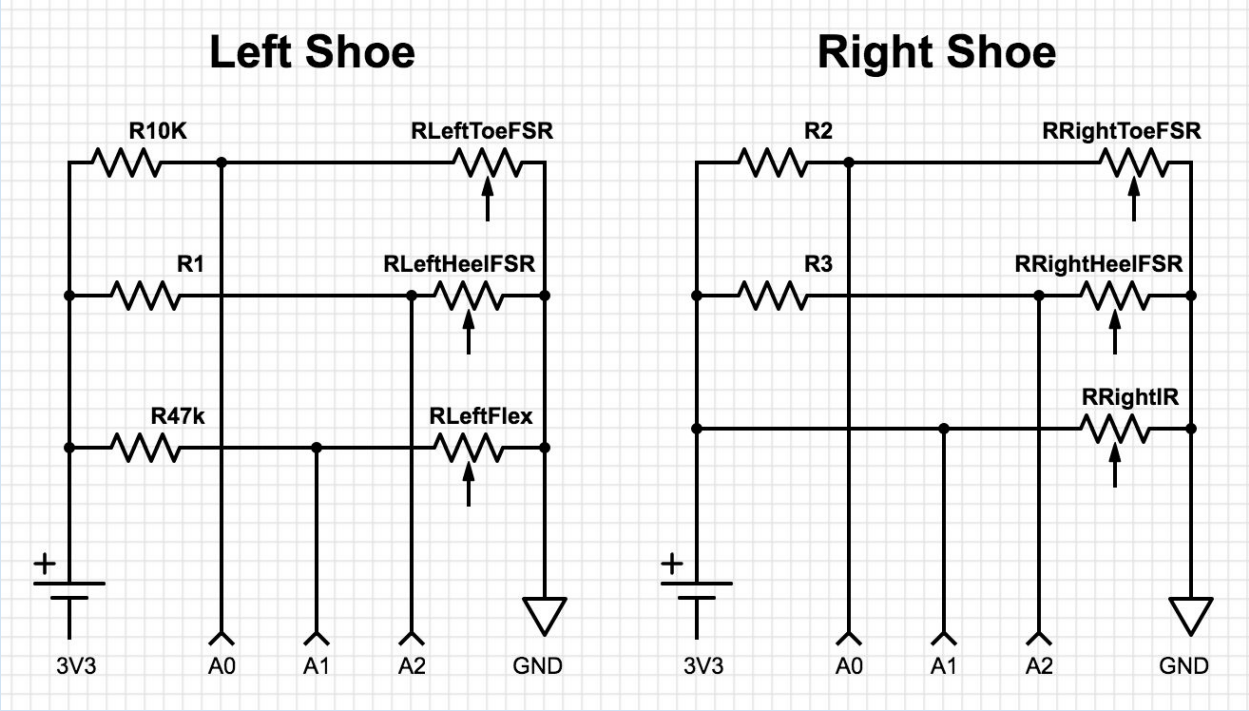
3.4 Block Diagram

Below is a high-level view of the main components and how they interact, with the low-level circuitry omitted.



3.5 Low Level Circuitry

Below is a low-level view of the circuitry. Any component whose label begins with R is a resistor, whether variable or standard. The labels along the bottom all correspond to ports on the relevant microcontroller.



4. Construction

4.1 Enclosure

The enclosure for *Shoes* is a modified pair of high-top sneakers. High-tops are necessary to allow sufficient room at the heel of the shoe for the microcontroller, power shield, and battery. The shoes needed to have sufficient structural integrity to allow for a number of small holes to be made for wires. A fairly inflexible exterior was preferable to prevent unintended movement (bouncing, drifting) of the IR sensor. For this purpose, I selected a pair of knock-off Adidas high-tops, which filled the above requirements.

4.2 Component Housing

For now, the microcontroller bundle is attached directly to the outer heel of each shoe (Fig 1, below). However, eventually each shoe will have a thin plastic container attached to the outer heel, inside which the Photon, power shield, and 3.7V Li-Po battery will be housed. The microcontroller bundle has a USB input on the power shield to allow for

USB recharging, as well as a number of ports on the Photon to which stranded-core single-conductor cables from the potentiometers are connected.
The force sensitive resistors are attached to the toe and heel of each shoe (Fig 2, below) with high-adhesive velcro pads, to allow for easy replacement of faulty sensors. The wiring is routed through holes in the base of the shoe next to each FSR, through the interior, and through another small hole in the heel leading to the microcontroller.
The IR sensor is attached with hot glue to the inner heel of the right shoe (Fig 3, below), near the base, pointing at the left shoe. The wiring is routed through a small hole in the heel of the shoe, through the interior of the heel, and through another small hole in the outer heel leading to the microcontroller.
The flex sensor is attached with hot glue to the top of the left shoe, near the toe (Fig 4, below). The wiring is routed under the laces into the interior of the shoe, and through a small hole in the outer heel leading to the microcontroller.
All of the sensors are wired with detachable cables (Fig 5, below), with pins at each end of the cable and stackable headers attached to the sensors and the microcontroller. This is in the interest of future-proofing, to allow for easy replacement of these cables if need be.

Fig 1. Microcontroller bundle attached to outer heel

Fig 2. Force Sensitive Resistors attached to toe and heel (holes in base of shoe obscured)

Fig 3. IR Sensor attached to inner heel of right shoe

Fig 4. Flex sensor attached to top of left shoe, near toe

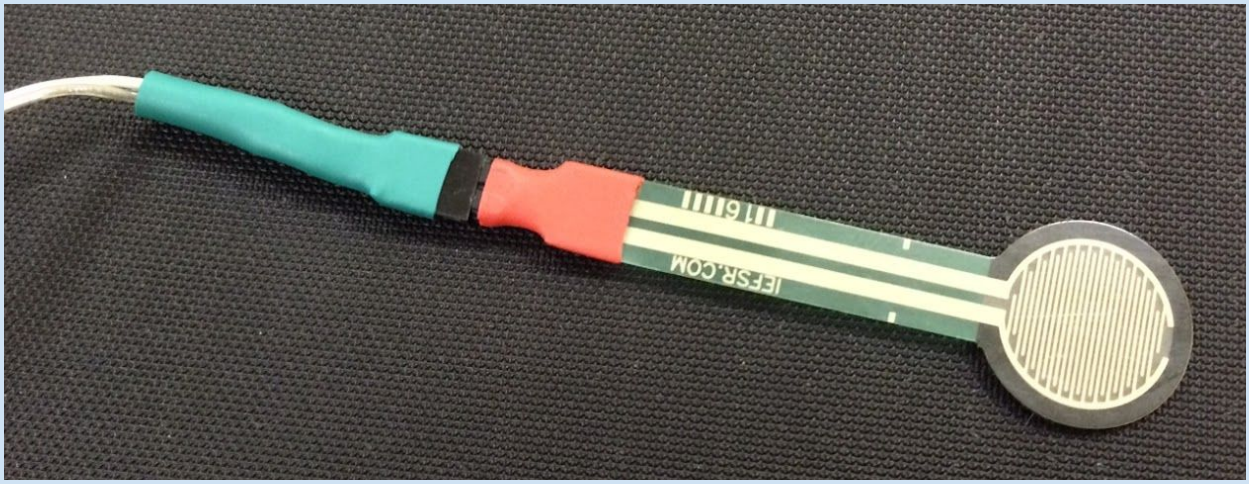


Fig 5. Detachable sensor connection, seen here with an FSR

5. *Shoes* In Use

5.1 Solo Instrument

Shoes is able to be used as an expressive performative instrument, capable of being a soloist in a group, but with in-built loop functionality to allow for layered performance outside of a group setting.

In this context, I have found *Shoes* to be easy to learn, but hard to master, with a learning curve allowing for beginners to be quickly engaged, while experts can still develop a depth of skill with practise.

I am yet to fully implement the software for *Shoes'* Solo Instrument Mode, so at this point there is only one performance style/sound. Thus, the below notes about some parameters having 'two modes' do not yet apply, but I fully intend to implement this functionality in future.

While there are many ways in which the various parameters could have been used to achieve this use case, my own implementation uses them thus:

NYI = Not Yet Implemented

- IR Proximity Sensor: Distance between feet used to control pitch. Two modes: continuous and *discrete (NYI)* - think fretless and fretted.
- Right Toe FSR: Used to trigger notes, and affect the volume thereof. Two modes: pressure and tap (tap detection implemented, *use of tap to generate sound natively NYI*). Pressure allows for highly expressive volume swells, tap detects spikes in force to provide note on/off messages.
- Right Heel FSR: Instrument select *(NYI)*. Taps will cycle through 'instrument' types, i.e. different sounds and the playstyle modes (see IR and Right Toe FSR modes above, and Flex Sensor modes below) associated with each sound.
- Flex Sensor: Used to affect sound being played. Two modes: *pitch bend (NYI)* and vibrato.
- Left Toe FSR: Loop control and Pitch Lock. Tap for loop-start, tap again for loop-end, tap while loop is playing for over-dub. High pressure on this FSR locks the pitch of the instrument so that use of the flex sensor does not create wild jumps in pitch.
- Left Heel FSR: Loop delete. Tap once to stop playback, hold to delete all.

5.2 Performance Augmentation

Shoes is also able to be used to affect other sound, augmenting the performance of a musician playing another instrument. In this context, *Shoes* is intended to be highly flexible, with it being up to the user as to how the parameters will be utilised, in much

the same way as the Monome. However, I have provided a ‘default’ Max patch to demonstrate its capabilities in this realm; a live harmoniser, which harmonises with input audio (such as a voice or saxophone) on the fly. The parameters are utilised thus:
NYI = Not Yet Implemented

- IR Proximity Sensor: Pitch control (interval). Distance will determine how far above/below the input audio’s pitch the harmony will be.
- Right Toe FSR: Harmonisation on/off. Holding down the toe will engage the harmoniser. Pressure allows for expressive volume control.
- Right Heel FSR: Harmonisation above/*below (NYI).* Tapping switches between the harmony being a high-part or a low-part.
- Flex Sensor: *Pitch bend (NYI).* Flex will allow for small modulation of pitch, either for fine-tuning or expressive purposes.
- Left Toe and Heel FSR: Unused

5.3 Video Demonstration

This is a video demonstration of the Solo Instrument (both native sound generation and external sound generation) and performance augmentation modes of *Shoes*.
Most aspects of *Shoes* performed very satisfactorily during this demonstration. I was pleasantly surprised by how finely and expressively I was able to control both pitch and volume - it really did feel like an ‘instrument.’
As discussed in 6.1 - Problem Solving, battery life posed a problem, with the batteries on each shoe dying twice during the course of the preparation and filming.
Unfortunately, a CPU issue stemming, I believe, from the interaction between Max/MSP, Soundflower, and Reaper (the latter two of which were used to record audio but are not necessary for general performance) meant that I was unable to demonstrate looping, despite it being fully implemented in my code.
A disclaimer before watching; *Shoes* is an instrument, and as mentioned above is ‘hard to master.’ The interface having only existed for a week and a half prior to filming, I have certainly not mastered it! As such, this is only a very basic demonstration of its functionality.

5.4 Potential Use Case: Software Control

A purpose for which I will not personally be using *Shoes*, but for which it would be well suited if a user should wish to implement it, would be controlling software, such as a DAW, outside of a live context. In this case, *Shoes*’ many parameters could be used in composition to increase ease of use of the software in question. Once again, this could

also be useful in allowing people with upper-body disabilities, who would otherwise find it difficult to use this kind of software, to engage with this method of composition.

6. Future Work

6.1 Problem Solving

A number of unforeseen issues arose during the development of this prototype, of varying degrees of severity. These issues, and how I might approach them, are detailed below.

- *Noisy IR Data:*
  Despite attempting to minimise sensor noise from the IR proximity sensor as much as possible through secure attachment and basic smoothing, the IR data remains fairly noisy. I think the best approach here would be to use modelling, perhaps something like a Kalman Filter which could provide very smooth data without sacrificing responsivity.
- *Deterioration of FSRs:*
  It seems that the consistent weight of a person being applied to the FSRs is causing some crushing of the sensors, reducing their data range drastically, and eventually rendering them unusable. While this can be addressed by simply replacing the FSRs regularly, a more satisfactory solution would be using a variety of FSR more suited to heavy weights, such as those used in digital scales ('load sensors'). This would give the added benefit of highly accurate data.
- *Short Battery Life:*
  In order to reduce performer encumbrance as much as possible, I chose to use a very small 100mAh LiPo battery, believing this would be sufficient to power a small device like the Photon. It turns out that Photons are quite power-hungry, and my current batteries only give about half an hour of battery life. An obvious solution would be to use a larger battery, but I would rather not increase the weight of the devices any more than necessary. As such, I think I will pursue a firmware solution here, perhaps implementing a sleep mode.

6.2 Improvements

As mentioned earlier, the current iteration of *Shoes* is very much a prototype, and there are a number of improvements I would like to make to it. These are listed below, with my current ideas as to how to approach them:

- *Implement Linear Pitch Response:*
  Currently the values returned by the IR sensor, like all IR sensors, are exponential, meaning that the pitch response is not particularly intuitive for the performer, and making fine pitch control *very* difficult at higher pitches. This could

be fixed by applying logarithmic correction to this data, to counteract this exponential scale. Doing so would give a far more predictable and usable pitch response over the full range of movement.

- *Interpolate Pitch:*
  As the Photons are only sending packets once every 5ms, very large, quick changes in foot proximity result in not so much a pitch sweep, as a series of pitch jumps. Interpolating pitch when these large, quick changes occur would allow for wide, smooth pitch sweeps.
- *Sensor Fusion*:
  As mentioned above, the IR data is quite noisy. Also, as a result of using IRs, pitch control is dependent on there being a clear sightline between the two shoes. I would like to improve upon this approach by exploring sensor fusion - specifically attaching an accelerometer to each shoe to get an idea of its place in space relative to its starting position, and melding this data with the IR data. This could potentially result in far more accurate and less noisy proximity detection, as well as possibly allowing for pitch control without a direct sightline between feet at all times.
- *UDP Multicasting:*
  In the current implementation, the IP address of the sound-generating computer is hard coded directly into each microcontroller's firmware. This is a perfectly reasonable approach, however I would like to explore the possibility of *multicasting*, an approach to UDP that eschews this hardcoded IP address. Instead, the UDP transmitting device simply outputs a signal in all 'directions' that any other nearby device can then tap into. This would allow the data streams being output by *Shoes* to be used by *any number* of nearby devices, running anything from sound generation programs, to video, to statistical data gathering, simultaneously. This would greatly increase *Shoes'* performative potential!