

# A position sensing midi drum interface

**Andrew Baxter**

64 Blades Street  
Lancaster, LA1 1SQ  
andy@ganglion.me

## Abstract

This paper describes my attempts at making a cheap, playable, midi drum interface which is capable of detecting not only the time and velocity of a strike, but also its location on the playing surface, so that the sound can be modulated accordingly. The design I settled on uses an aluminium sheet as the playing surface, with piezo sensors in each corner to detect the position and velocity of each strike. I also discuss the electronics (arduino based), the driver software, and the synths I have written which use this interface.

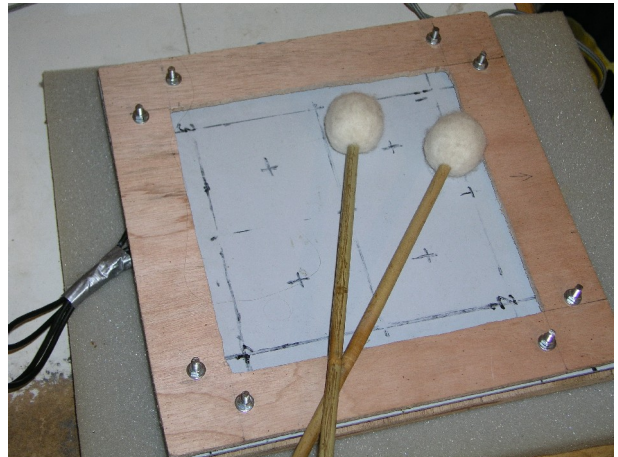
## Keywords

Drum interface, MIDI, Position sensitive, Supercollider, DIY

## 1 Introduction

Midi drum interfaces are now a widely available piece of consumer electronics. However, in most cases they are only capable of reproducing a small range of the sounds you can make using a real drum. A significant weakness of these interfaces is that they do not give any indication of where on the playing surface you have struck them, so they are limited to playing a single sample or synthesised sound across the whole surface.

In this paper, I will describe my attempts at making a cheap, playable, midi drum interface which is capable of feeding information on where you have struck it to a synthesiser, which can modulate the sound accordingly, in two dimensions. There already a few devices coming out at the high end of the market which have similar abilities; however my aim here (apart from having a bit of fun myself building it), was to produce a design which is simple and cheap



*Illustration 1: The pad from above*

enough for any competent hobbyist to build, using widely available components.

## 2 Research

Before settling on the current design, I tried out a couple of other ideas for how to make such a drum pad.

My original idea was to use a sheet of conductive rubber as the playing surface, with a voltage put across it alternating between the north-south and east-west axes, so that a strike at a given point would correspond to a particular voltage pair. The simplest form of this idea would require the sticks to have wires on them, so is not really practical, but I discovered that there is a form of conductive rubber which lowers its conductivity sharply under pressure. This gave me the idea of making a sandwich with the voltage gradient sheet on the top, then a layer of pressure sensitive rubber, then an aluminium sheet electrode under both. Strikes to the top surface would, I hoped, produce small regions of lowered conductivity in the middle sheet, transferring the voltage at that point from the top sheet to the bottom electrode.

I constructed a prototype of this design, and managed to get it to sense position to a degree, but

decided in the end that the results weren't consistent enough to be worth carrying on with this idea. Also, this design was fairly expensive (due to the cost of the pressure sensitive sheet), and lacked a reliable way of sensing the velocity of a strike.

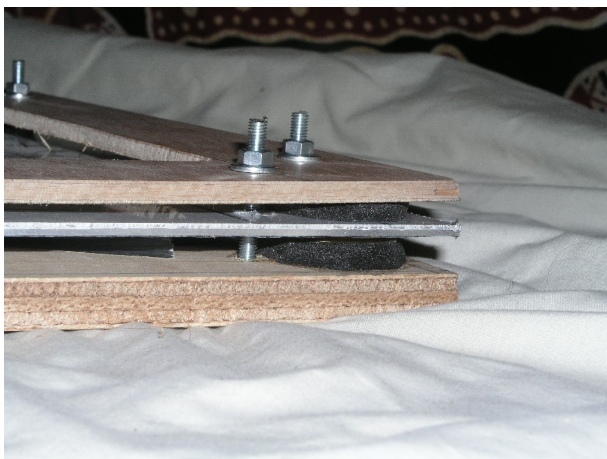
The next idea, which I owe to Alaric Best, was to suspend a metal sheet in some kind of frame with vibration sensors around the edge, and detect the time of flight of pressure waves from the strike point to each sensor, and use this to triangulate the position of the strike.

I built a prototype of this using piezoelectric sensors, but was unable to get the sensors to detect pressure waves at anywhere near high enough time resolution.

However, during the testing, I noticed that the strength (rather than the timing) of the signal from the piezo sensors varied according to how close the strike was to each sensor (with one sensor in each corner of the sheet). In other words, on the time scale I was able to sense at, the piezos were simply detecting the transferred pressure of the strike at each mounting point. This gave me the idea for the current design.

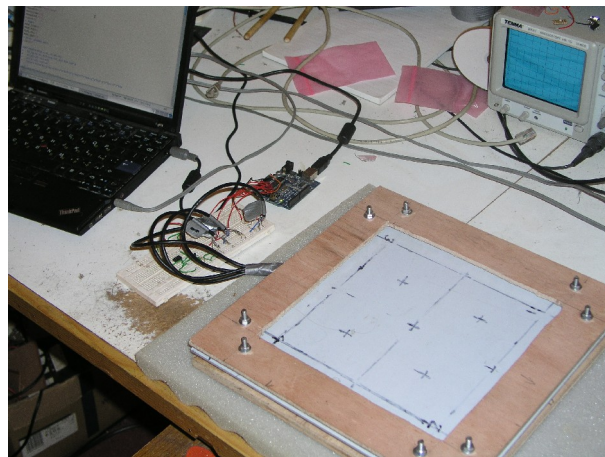
### 3 Design and construction

The current physical design of the pad is as follows (see also illustrations 2 and 3):



*Illustration 2: Corner view of the pad, showing rubber buffers and bolts. You can also just see the edge of the piezo sensor on the lower buffer.*

- The playing surface is a square sheet of aluminium.
- This is suspended between foam rubber buffers in a wooden frame. The buffers



*Illustration 3: The pad connected to the driver circuit and a laptop*

support the sheet above and below at each corner.

- The two halves of the frame are held together by bolts near each corner.
- Holes drilled in the sheet allow the bolts to pass through the sheet without touching it, so that it can move freely with respect to the frame.
- Under each corner of the sheet is a piezo-electric sensor, mounted above the lower buffer, in such a way that all the pressure from that corner of the sheet is transferred through the sensor. The distribution of strike pressure between these sensors indicates the position of the strike.
- Coax cables are used to bring the signals from the sensors out to a circuit board, where they are detected by an Arduino microcontroller board and fed back to a computer through usb bus.
- The whole thing rests on a soft foam rubber pad to reduce the effect of vibrations.

Full instructions on how to build one of these pads are available on the web.<sup>1</sup>

### 4 Electronics

The electronics for the pad are pretty straightforward – I used an arduino microcontroller board to detect the voltage pulses from the piezos; the only additional electronics was a simple voltage source to provide a false

<sup>1</sup><http://www.instructables.com/id/A-position-sensitive-midi-drum-pad/>

ground at half the arduino's 5v analogue input range. This allows the board to detect negative going as well as positive going pulses from the piezos. The voltage range from the piezos matches the arduino's analogue input range well enough that no additional amplification or attenuation is needed in practice.

The schematic for the circuit is shown in illustration 4.

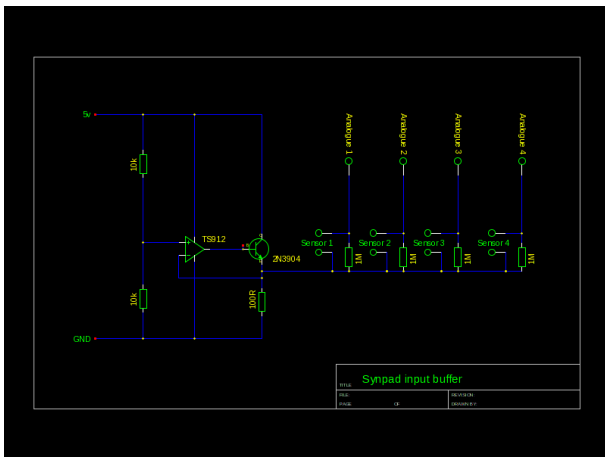


Illustration 4: Schematic for the pad's input circuit.

## 5 Driver software

The driver software for the pad is in two parts – a small firmware program on the arduino, which feeds basic strike data back to a laptop through its usb cable, and a longer program on the laptop which calculates the position and velocity of each strike from this, and sends this information to a software MIDI channel. All the software is available on the web<sup>2</sup>.

### 5.1 Arduino firmware

The arduino firmware works as follows:

- On startup, the four analogue inputs are read and the base readings stored.
- The four analogue inputs are then read every 100 us. The base reading for each input is subtracted to give the signal level.
- If the signal level on any input exceeds a trigger level, then the program starts a measurement cycle.

The measurement cycle goes like this:

- The signal levels on each input are read every 100us for a set number of readings (currently 10).
- At each reading, the absolute value of the signal level on each input is added to a sum for that input.
- At the end of the measurement cycle, the summed values for each sensor are sent as a comma separated text string back to the laptop for further processing.

There is then a delay (currently 30 ms) to prevent re-triggering on the same strike, after which the program starts waiting for the next strike.

### 5.2 Midi mapper

The raw data from the arduino is then interpreted by a python program on the laptop (the midi mapper).

There are two phases to using this program. First, it needs to be calibrated with a set of strikes at 13 known positions on the pad (which are marked on the playing surface, as you can see in Illustration 1). The raw sensor values and known x-y position for each strike are recorded in an array.

```
def mapCurve(p,s1,s2,s3,s4):
    # p is an array of 7
    coefficients; s1..4 are the raw
    sensor readings
    k2,k3,k4,l1,l2,l3,l4=p #
    give names to the coefficients.
    # the k coefficients allow
    for different sensitivities of
    the sensors.
    f1=s1 # first k coefficient
    is always 1
    f2=s2*k2
    f3=s3*k3
    f4=s4*k4
    # the l coefficients allow
    for irregularities in the
    physical construction of the pad
    x=(l1*f1+l2*f2+l3*f3+l4*f4)/
    (f1+f2+f3+f4)
    return x # the mapped
    coordinate (either x or y)
```

*Text 1: Python code for the position mapping equation*

<sup>2</sup><http://ganglion.me/synpad/software/>

Once the last calibration reading has been taken, these readings are used to fit a simple equation which is based on a rough physical model of the pressures transferred through the sheet. The code which implements this equation is shown in the frame 'Text 1' above. This is done separately for the x and y coordinates, producing two sets of coefficients which can be used to turn incoming strike data into x-y coordinates. The algorithm used to fit the data is the least squares function from the 'scipy' python library<sup>3</sup>.

In the second phase, the coefficients from the calibration are used with the equation to determine x-y coordinates for live strike data. These are then sent as a stream of midi events on a software midi channel to a synthesiser. The way the coordinates are encoded into a midi stream is as follows:

- first, 'set controller' events are sent on controllers 70 and 71, corresponding to the x and y coordinates.
- then, a 'note on' event is sent, with the velocity proportional to the sum of all the raw sensor values, and the note number equal to the x coordinate.

This information can then be used by a synthesiser to create a sound which varies according to the x, y and velocity coordinates of the strike.

## 6 Sound synthesis

In principle, the midi stream for the pad could be fed into any drum-like soft-synth that is capable of modulating the sound according to midi controller values. (By drum-like I mean that the synth should not require a note off event to end each strike sound.)

However, in practice I decided to use the Supercollider audio synthesis language<sup>4</sup> to construct the synths for the drum. Other similar environments, such as csound or pure data, could have been used, but supercollider seemed to offer the greatest level of control and flexibility, and suits my way of thinking as a programmer. (Once I had got my head round its syntactic quirks!)

The supercollider code I have written is available on the web<sup>5</sup>. It is in two parts – the first

```
SynthDef.new("MidiDrum", { |
vel=100, x=64, y=64,out=0|
  // synth drum with pink
noise, comb delay line and low
pass filter.
  var rq=10**((y-40) / 41);
  var env,amp;
  var noteMin=55; // 200Hz
  var noteMax=128;//66;
  var      note=(x*(noteMax-
noteMin)/127)+noteMin;
  var baseFreq=100;
  amp=16*((vel-96)/3).dbamp;

env=EnvGen.kr(Env.perc(0.01,0.5,
1),1,doneAction:2);

Out.ar(out,amp*env*Pan2.ar(LPF.a
r(CombC.ar(PinkNoise.ar(0.1),1,1
/baseFreq,rq),note.midicps),
0) );
} ).store;
```

*Text 2: A sample synthdef for the drum (written in supercollider)*

(drummidi.sc) listens for midi events on a channel and uses them to trigger a synthdef with the right x and y parameters. The second part (synpad.sc) is a set of synthdefs which have been written for this interface. This code is still in a pretty crude state, which works well enough for experimenting with different sounds, but wouldn't really be suitable for a live performance situation.

One of the synthdefs I wrote is reproduced in frame 'Text 2'. It takes 3 variable parameters – the velocity and x-y coordinates – and converts these into a percussive sound whose timbre varies across the pad.

## 7 Results

In this section I will write about how the drum performs in practice, starting with the physical construction, then looking at the interface's playability, accuracy and ease of use, and finally discussing the synths I wrote to play it through<sup>6</sup>

<sup>3</sup><http://www.scipy.org/>

<sup>4</sup><http://supercollider.sourceforge.net/>

<sup>5</sup><http://ganglingion.me/synpad/software/>

<sup>6</sup>A video of the pad in use is available on the web. See <http://ganglingion.me/synpad/> for a link to this.

## 7.1 Physical construction

Actually building the pad was fairly easy. All you need to make one are a few cheap materials, some very basic carpentry and metalwork skills (cutting and drilling wood and aluminium sheet), and some simple tools. Mounting the piezos and soldering the contacts to them was a bit awkward, but no more than that.

Similarly, the electronics construction skills you need are pretty minimal, as the arduino board is doing most of the work.

## 7.2 Playability

The pad is fairly easy to play, with either fingers or felt headed timpani sticks. The biggest problem is the height of the frame above the board, which can make it a bit awkward to reach the edge of the playing surface.

## 7.3 Accuracy

The accuracy of strike detection is good enough to get some reasonable results. The pad senses velocity pretty well, although there is a lower cutoff which makes it hard to play very soft notes. The consistency of the position sensing is not bad – if you hit the pad repeatedly in the same spot, the position stays constant with velocity to about 10-15%<sup>7</sup>. There is a degree of distortion in the mapping between pad positions and detected coordinates, but this error is not so much of a problem in practice, as you can adapt your playing to compensate.

Drift from the calibrated mapping during playing is small enough not to be a problem in practice. The pad would probably need recalibration at the start of a performance though, especially if it had been handled roughly during transportation.

In the time domain, the triggering delay (latency) imposed by the arduino firmware is about 1msec. I have not tried to measure the latency of the midimapper program, but in practice the latency of the combined system (firmware plus midi mapper plus synths plus sound card latency) is good enough that the strike sounds appear immediate to my ear.

---

<sup>7</sup>N.B In the original paper sent to the LAC 2011 organisers, I had a value of 5% here, which I have now corrected after measuring it again.

## 7.4 The synths

Writing modulateable synths which sound good has proved to be the most difficult part of this project. The first thing I tried was to play a sample of a snare drum through a resonant low pass filter, with the x-coordinate controlling the filter cutoff, and the y-coordinate controlling the resonance. This produces some interesting effects, and is fun to play with. The drawback is that it is hard to make strongly rhythmic patterns with it: because the filter is resonant and the cutoff varies quite rapidly across the pad surface, it's hard to hit close enough to the same point to repeat a given sound consistently – the sounds appear to the ear like a series of separate tones rather than variations of a single sound.

My next idea was to make something that was based on a more consistent base tone, with the strike coordinates modulating its timbre. This is the synthdef reproduced in frame 'Text 2' above. It is based on pink noise filtered through a comb delay line and then a non-resonant low pass filter. One coordinate controls the resonance of the delay line (the comb frequency is fixed), and the other controls the cutoff frequency of the low pass filter. This produces a nice synthy sound, with the timbre varying from noisy to ringing in one dimension, and from muted to bright in the other. This is the synth I used for the online demo video of the drum<sup>8</sup>.

Some other things I tried:

- working through the percussion section of the 'Synth Secrets' articles from Sound On Sound magazine<sup>9</sup>. I managed to make some half decent percussion sounds like this (though cymbals are tricky). The difficult part was more in working out a meaningful way of modulating the sound across the pad. Because these synths have many variables, any of which could be used as modulation parameters, it's hard to decide what combination of variables to vary to get a nice result.
- Feeding audio samples into an FFT and operating on them in frequency space in

---

<sup>8</sup>See <http://ganglion.me/synpad/> for a link to the video.

<sup>9</sup>See <http://www.soundonsound.com/sos/allsynthsecrets.htm>

various ways. I was hoping that this would produce a series of modulateable effects which could be applied to any base sample, making for a rich palette of sounds. However the results were a bit disappointing, probably due more to my lack of experience with supercollider and audio synthesis in general than anything else.

Overall, I think the basic concept of modulating a synthesised sound across the playing surface is good, and I've enjoyed writing and playing with some simple synths. At the same time, I've also come to realise that there's a lot more to writing synths from the ground up than I had originally thought, and producing a range of effects good enough for live performance could involve a fair amount more work.

## 8 Similar work

In this section, I mention a few projects / products which are working in a similar space.

### 8.1 Korg Kaoss Pad<sup>10</sup>

This is superficially similar to the pad I have made, in that it has a square playing surface which you can use to control sounds in 2 dimensions. However its function is quite different – it doesn't have a velocity sensing function and its role is as an effects processor for sounds generated elsewhere, rather than an instrument in its own right. It sells for around 300 USD.

### 8.2 Mandala Drum from Synesthesia Corp<sup>11</sup>

This has a circular pad with 128 position sensing rings arranged concentrically on it. It can only modulate the sound in one dimension rather than two, but the design appears to be much more polished and playable than mine. It is also sold with a library of sound effects tailored for the drum, some of which emulate the sound of a real snare drum. They sell for about 350 US dollars.

### 8.3 Randall Jones's MSc thesis on 'Intimate Control for Physical Modelling Synthesis'<sup>12</sup>

This uses a 2D matrix of copper conductors arranged perpendicularly on either side of a rubber sheet. Each north-south conductor carries a signal oscillating at a different frequency, and the east-west conductors pick up these signals by capacitive connection, to an extent which varies according to where pressure has been applied to the rubber sheet. The signals are generated and received by a standard multi-channel audio interface, and interpreted in software on a computer.

This project is probably the closest to mine in its intent – it's a midi drum surface with two dimensional position and velocity sensing. It has also been designed in a way that most people could build one themselves. As far as playability goes it looks to be way ahead of mine – it is multitouch, can detect continuous pressure changes as well as instantaneous strikes, and the profile of the frame around the head is lower, which should make it more comfortable to play. It is also self-calibrating, so doesn't need to be set up again every time you play.

Its main drawback is complexity and the associated cost. There is a lot of signal processing going on to produce the admittedly impressive result. The fact that it depends on a separate sound card also makes it fairly expensive compared to my project.

## 9 Improvements and future directions

Here I discuss ideas for where I might take this project in the future.

If I was to stick with the current basic design, there are a few simple improvements I could try, to make it more playable and responsive. For example, instead of a single wooden frame, the aluminium sheet could be held in place by metal discs bolted to the base board at each corner. This would make it easier to reach the pad surface when playing.

There might also be small improvements possible in the firmware and the midimapper, to

---

<sup>10</sup>[http://en.wikipedia.org/wiki/Kaoss\\_Pad](http://en.wikipedia.org/wiki/Kaoss_Pad)

<sup>11</sup><http://synesthesiacorp.com/about.html>

---

<sup>12</sup>[http://2uptech.com/intimate\\_control/](http://2uptech.com/intimate_control/) Video here: <http://vimeo.com/2433260>

improve the consistency and accuracy of the position sensing.

However, since having looked at Randall Jones's design, I'm thinking that this is much closer to the direction things ought to be going. So, in the future I would be more interested in developing something which offers a similar degree of responsiveness and playability, either by adapting his design to make it simpler and cheaper to build, or using some other technique.

I also have a few ideas about developing the associated software to make it more powerful and easier to set up and use. For example, it wouldn't be hard to build a graphical interface which would let you swap between different synths, rather than having to evaluate supercollider code to do this, as at present.

One idea I would like to have a go at is to make some synths with several variable parameters, then find a way of assigning parameter-sets (presets) to different points on the pad's surface. It should then be possible to use some kind of mapping algorithm to smoothly vary the parameters of the synth across the pad's surface in such a way that at each preset-point, the result sounds like the preset you have assigned to that point, and in between points the sound smoothly morphs from one preset to another.

## 10 Conclusion

In conclusion, I think that the basic concept of creating a 2 dimensional playing surface for synth percussion sounds is sound, and has a lot of potential. I have been fairly successful in achieving my aim of making such a surface using cheap, simple components. However, this particular design has a number of flaws, such its lack of multi-touch and continuous pressure sensing abilities, the need for calibration, and a degree of physical awkwardness in playing it, due to the height of the mounting frame.

I am planning to continue developing the idea, and may put some more work into refining this design, but in the long run something like Randall Jones's design looks like a better way forward for this kind of interface.

On the software side, the hardest part is producing good synth sound effects for the pad. Because this kind of interface is quite new, it is necessary to write new synths for it from the

ground up rather than using existing ones. There is also a lot of room for improvement in the supporting software more generally, and I am planning to put some more work into this in the future.

## 11 Acknowledgements

Thanks go to Alaric Best and Dave Leack of Veraz Ltd. for their thoughts on and encouragement with the project. Also to the writers of the Supercollider audio synthesis language, without which the task of writing a synth for the drum would have been ten times harder.

## References

- [1] R. Jones. 2008. Intimate Control for Physical Modelling Synthesis. *MSc Thesis, University of Victoria*.