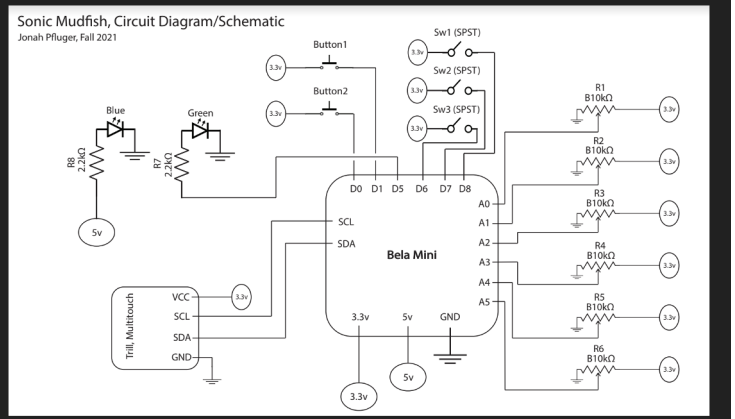
The *Sonic Mudfish*

Errant Machine Learning for Reactive Generative Synthesis

```
SynthDef.new(\n1, {
        arg in1 = 1, in2 = 1, w1 = 0.1, w2 = 0.1, b = 1, bW = 0.1;
        var y, x, update, sig, n6AudioFeedback, scaler;

        sig = InFeedback.ar(~synBus, 2);
    update = InFeedback.ar(~updateBus, 1);
    scaler = AnalogIn.ar(0).linexp(0.0, 1.0, 0.1, 50.0);
        n6AudioFeedback = InFeedback.ar(~n6AudioOut, 1);
        n6AudioFeedback = n6AudioFeedback * scaler;
        in1 = In.ar(~derive1, 1);
        in2 = In.ar(~derive2, 1);


        w1 = w1 + update * in1; // delta rule update
        w2 = w2 + update * in2; // delta rule update

        y = (in1 * w1) + (in2 * w2) + (b * bW); // weighted sums
        x = 1 / (1 + y.neg.exp); // sigmoid computation

        sig = CombL.ar(sig + n6AudioFeedback * 0.5, 2, AnalogIn.ar(3).range(0.1, 1.7).lag(1) + x.lag(1), 5);

        Out.ar(0, sig);
        Out.ar(~n1AudioOut, sig);
        Out.ar(~n1out, x);
}).add;

SynthDef.new(\n2, {
        arg in1 = 1, in2 = 1, w1 = 0.1, w2 = 0.1, b = 1, bW = 0.1;
        var y, x, update, sig, n6AudioFeedback, scaler;

        sig = In.ar(~synBus, 2);
    update = InFeedback.ar(~updateBus, 1);
    scaler = AnalogIn.ar(0).linexp(0.0, 1.0, 0.1, 50.0);
        n6AudioFeedback = InFeedback.ar(~n6AudioOut, 1);
        n6AudioFeedback = n6AudioFeedback * scaler;

        in1 = In.ar(~derive3, 1);
        in2 = In.ar(~derive4, 1) / 2000;
        n6AudioFeedback = In.ar(~n6AudioOut, 1);
```



Jonah Pfluger

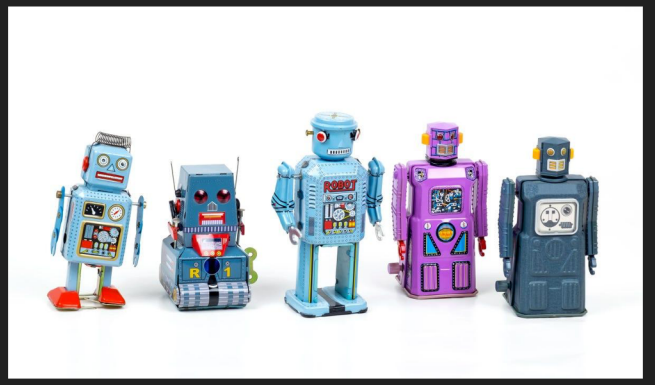
M.S. Music Technology Final Presentation
Fall 2021

Overview

- The *Sonic Mudfish* is a perceptron*-based synthesizer and audio processor designed for use in non-idiomatic electroacoustic improvisation
  - *the simplest form of artificial neural network (ANN)
- How it works (generally):
  - Machine listening and derived data -> excitation of a 2-3-1 perceptron algorithm
  - With this level of interactivity, the system alters audio effects parameters and generative music structures to gain a level of reactivity
- The *Mudfish* is built using a Bela Mini microprocessor running code written in the SuperCollider programming language

On Machine Learning and AI

- *Mudfish* is based on an algorithm used in artificial neural networks, but what are AI systems in general?
- Extremely wide variety of tools and technologies
- Common Connotation
- Russell & Norvig:
    - **Reactive Machines**
    - **Limited Memory Machines**
    - **Theory of Mind Machines**
    - **Self-aware Machines**

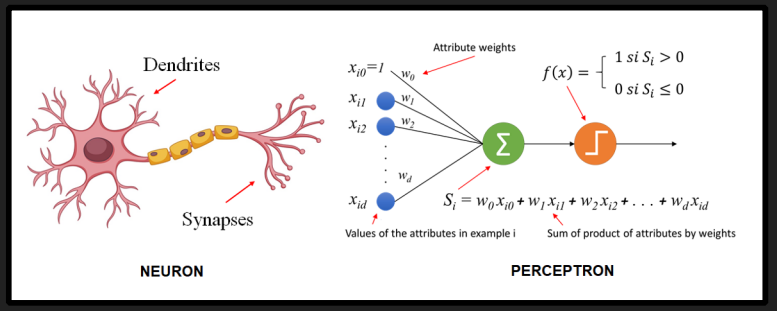


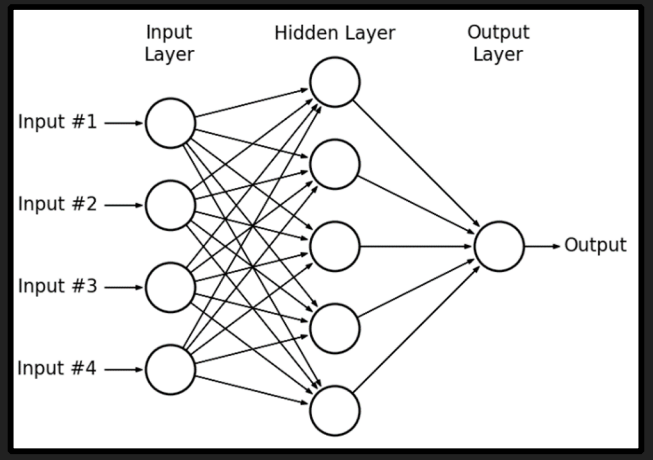
Intelligent systems v.s. computer programs..

- Can you test computational intelligence?
- With human intelligence still poorly understood, machine intelligence is often a highly metaphorical concept

Neural Networks & Perceptrons

- Function through node-based calculation of weighted sums
  - loosely related to the manner in which the human brain
  - *summed data = (input1 * weight 1) + (input2 * weight 2) + (input3 * weight3) + …*
- Activation Function:
  - *f(x) = 1 / 1 + e^-x*
- Parameter update: Delta Rule
  - *W(n+1)=W(n)+η[d(n)-Y(n)]X(n)*





Neuron Implementation in *Mudfish*

```
SynthDef.new(\n1, {
        arg in1 = 1, in2 = 1, w1 = 0.1, w2 = 0.1, b = 1, bW = 0.1;
        var y, x, update, sig, n6AudioFeedback, scaler;

        sig = InFeedback.ar(~synBus, 2);
    update = InFeedback.ar(~updateBus, 1);
    scaler = AnalogIn.ar(0).linexp(0.0, 1.0, 0.1, 50.0);
        n6AudioFeedback = InFeedback.ar(~n6AudioOut, 1);
        n6AudioFeedback = n6AudioFeedback * scaler;
        in1 = In.ar(~derive1, 1);
        in2 = In.ar(~derive2, 1);


        w1 = w1 + update * in1; // delta rule update
        w2 = w2 + update * in2; // delta rule update

        y = (in1 * w1) + (in2 * w2) + (b * bW); // weighted sums
        x = 1 / (1 + y.neg.exp); // sigmoid computation

        sig = CombL.ar(sig + n6AudioFeedback * 0.5, 2, AnalogIn.ar(3).range(0.1, 1.7).lag(1) + x.lag(1), 5);

        Out.ar(0, sig);
        Out.ar(~n1AudioOut, sig);
        Out.ar(~n1out, x);
}).add;
```

Broad Trends in Neural Synthesis

- Neural networks = great at extracting patterns from large sets of data
  - Less great generating meaningful predictions
- Application to music technology can be extremely varied
- Google’s NSynth Super
  - 4-axis vector-oriented synthesis
  - *pure research*: the integrity of an algorithm appears to outweigh its sonic implications
- David Tudor’s Neural Synthesizer
  - 80170NX implemented neurons as op-amps which could therefore be interconnected, much like gain stages in a no-input mixer
  - *errant machine learning*: an algorithm is fractured in favor of sonic intentions

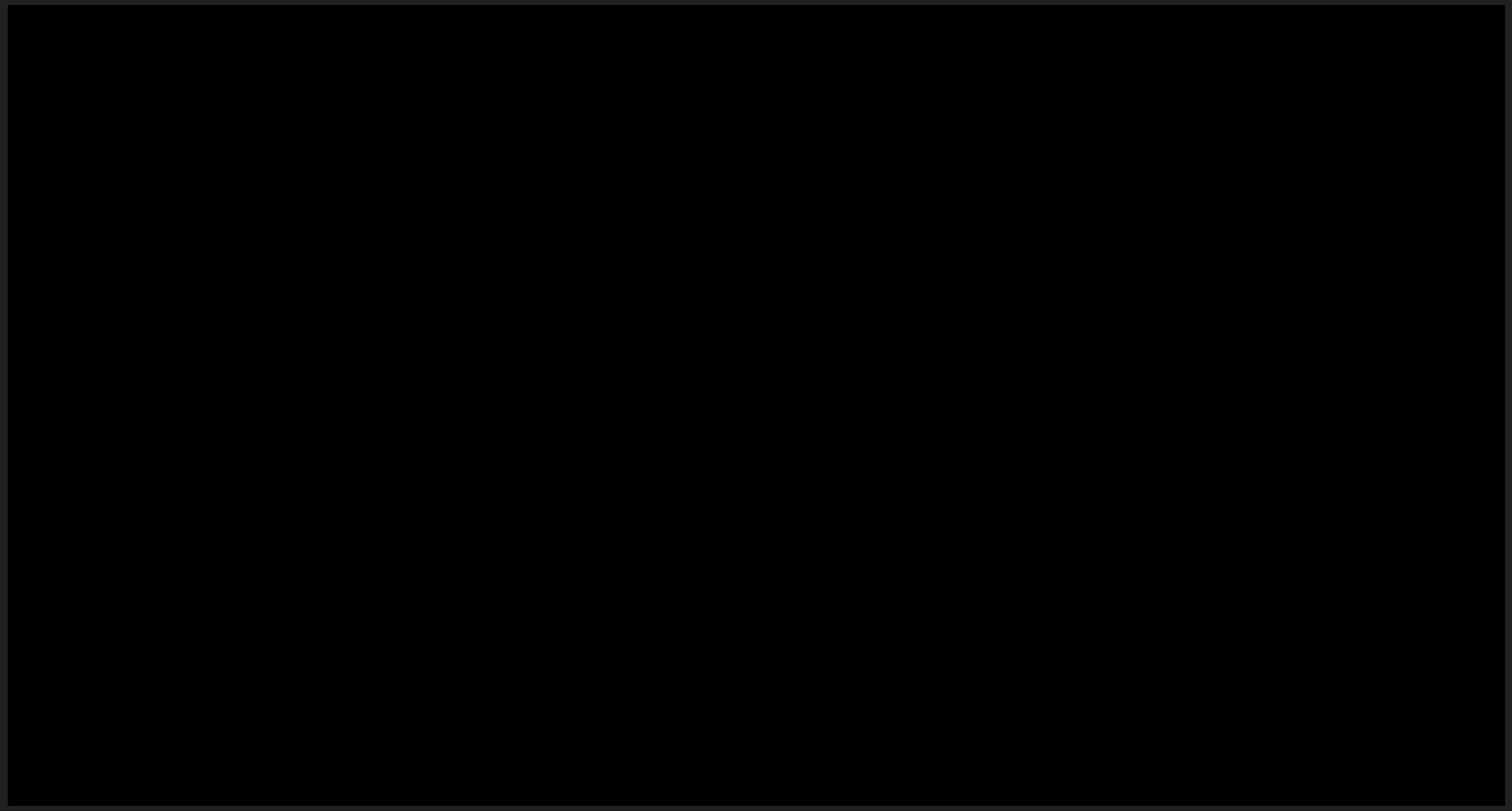
Artistic Goals

- Hardware-based interactive music system
  - “exhibit[s] changing behavior in response to human input”
  - Personalized approach to computational collaboration
- Derived sonic textures from input data
  - Reactivity via perceptron derivation
- Chaos, unpredictability, and the electroacoustic principle of “any sound as musical resource”
- Non-Idiomatic

Artistic Goals (cont…)

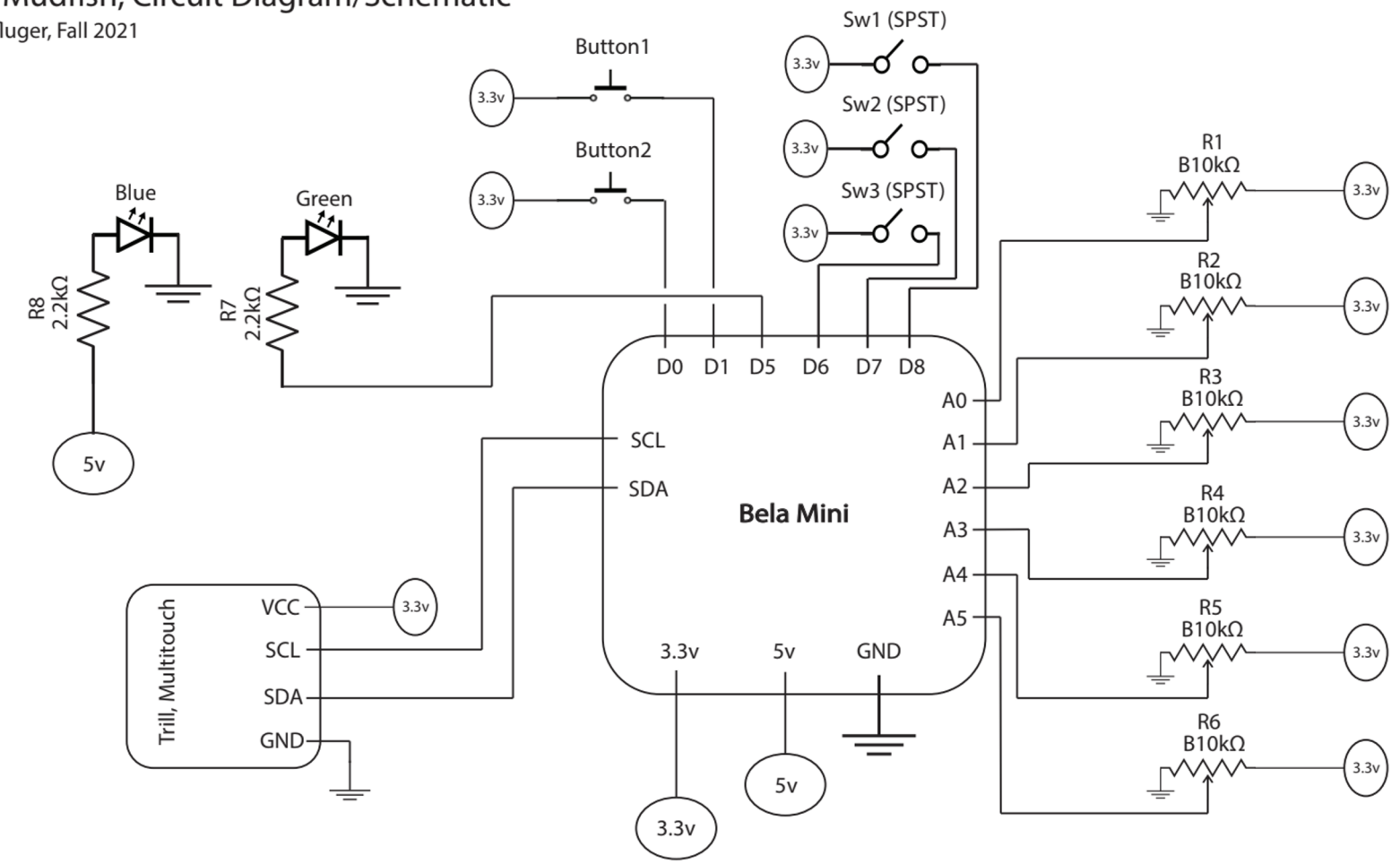
- Extending John Cage's assertion that sounds should be "allowed to be themselves", *Mudfish* experiments with the notion of allowing an algorithm to be itself through less musically-oriented mediation.
  - What can we learn from this?
- Post-digital approach to data sonification
  - Sonification - we learn from listening to data
  - Post-digital - we learn from breaking & reassembling technology

Core Functionality + Improvisation Excerpts



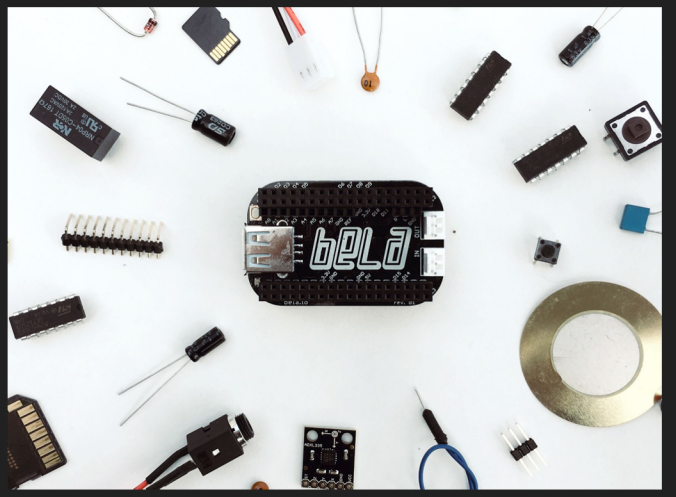
Sonic Mudfish, Circuit Diagram/Schematic

Jonah Pfluger, Fall 2021



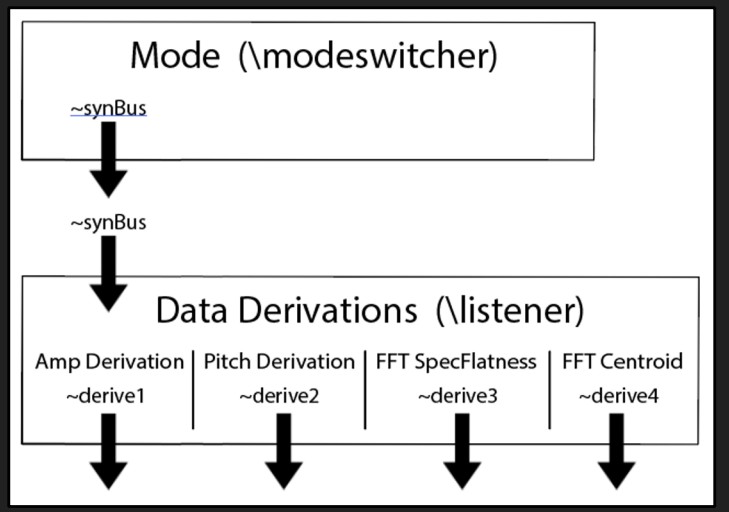
Bill of Materials

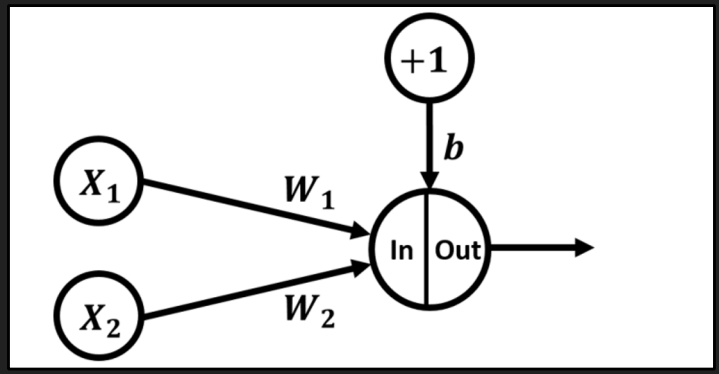
- Bela Mini
- 2 - ⅛” to 3pin Molex cables
- 6 - B10Ωk potentiometers
- 2 - 2.2kΩ resistors
- 3 - SPST switches
- 2 - SPST Buttons
- 1 - Trill Bar (multitouch slider)
- 2 - LEDs (one green, one blue)
- Lots o’ wire (solid core)
- Soldering supplies
- Acrylic (w/ M4 standoffs)



The Code… (essentially)

- Mode switch: “synthesized or external audio?”
  - Output result to machine listener
- Machine listener: “here you go!”
  - Amplitude analysis
  - Pitch Analysis
  - FFT Flatness
  - FFT Centroid
- Derivations excite perceptron
  - Audio and data through perceptron
  - SynthDefs as neurons
- Audio and data feedback through the perceptron
  - “No-input” functionality
  - Network as sonorus structure and control source





Reactive Generative Synthesis

Synthesis Structure	Generative Control of the Synthesis Structure
arg out = 0, freq = 35, amp = 0.5, atk_tim = 0.01, dkay_tim = 0.2; **var** sig, env, pitch_env, in1, in2, mod, in3, in4; **var** button2 = DigitalIn.ar(7); **in1** = InFeedback.ar(~n3out, 1); **in2** = InFeedback.ar(~n5out, 1); **in3** = InFeedback.ar(~derive4); **in4** = InFeedback.ar(~derive2); **env** = EnvGen.kr(Env.perc(atk_tim * in1, dkay_tim * in2), doneAction: 2); **pitch_env** = EnvGen.kr(Env([0.5, 1.5 - in2, 0.2], [0.01, 0.2])); **mod** = Pulse.kr(0.1 * in3.range(0.1, 5), mul: in4.range(0.1, 5)); **sig** = SinOscFB.ar(freq * pitch_env, mod, mul:5)!2; **sig** = LPF.ar(sig, 2500); **sig** = sig * env; **sig** = sig * 0.01; **sig** = sig * button2; Out.ar(~synBus, sig * 1.5);	**dur**, Pwhite(0.5, 5, inf), **freq**, Pwhite(1, 10000, inf) * Pfunc({~ampDerive})

What *Mudfish* is, what it is not...

- Mudfish is a simple reflex agent based on adherence to the following characteristics:
  - Functions primarily through responding to current input stimuli
  - No long term memory beyond 1 control period (1024 samples) to manage feedback and parameter updating
- Despite utilization of the perceptron algorithm, *Mudfish* applies this algorithm in a notably non-standard way
  - ML-informed, not “intelligent”

Looking ahead

- Additional AI/ML techniques
- More complex neural network
- CPU limitations on the Bela Mini microprocessor
  - in the box approach?
- Upgraded the enclosure
  - Damaged by laser cutter
  - Open sides
- Guitar pedal, eurorack module, etc…
  - Technical considerations
- System-based Instrument Design
  - Instead of using systems (neural networks, mathematical formulas, sonified data, etc) as generative influences, can they be used as the instruments themselves?
- Commercially viable products
  - Expense and iterability