Step Response Input Board

STEP RESPONSE

Neils favourite sensors

can measure with it - resistance, capacitance, inductance, position, pressure, proximity, tilt, acceleration, humidity, touchpad, multitouch, ... loading

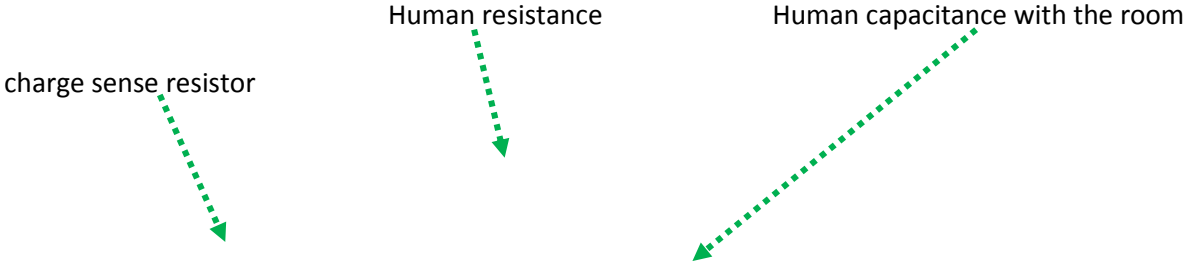
going to measure what's called the step response. We're going to look at 1 version with 1 electrode

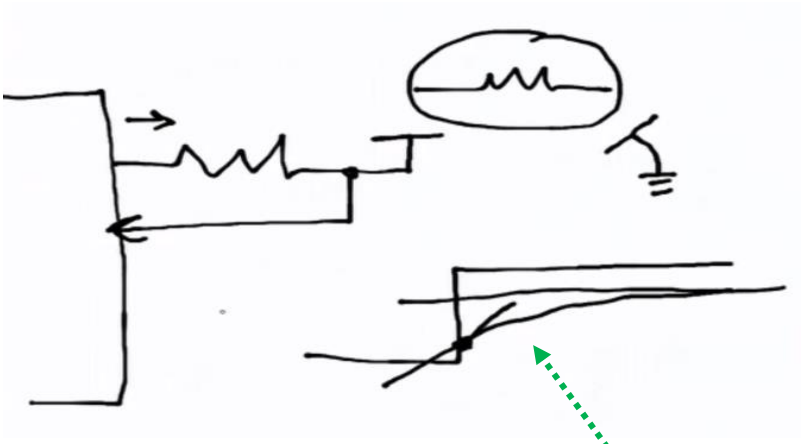
Step Response Example 1

We are using this as a proximity sensor, non-contact proximity sensor.

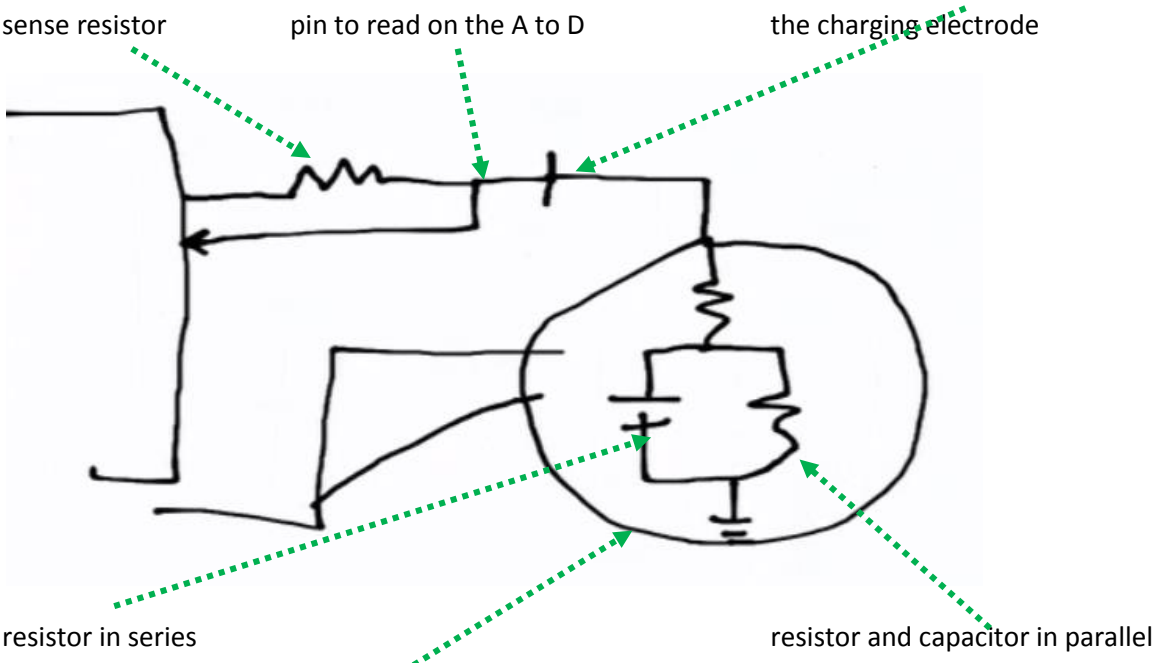
You have the processor and a resistor that's a current sense resistor. Going to talk out to that resistor but also going to listen here (that goes to the A to D) . In the loading version you just connect it to an electrode.

Internally a human is about a mega ohm resistance across your skin and then a kila ohm internally, your internal conductivity. We are pretty good conductors internally. We have capacitive coupling out to the room. All the things around you that are connected to ground means there is a capacitor from you to the room. This is typically 10s of pF's (picofarad). So what it does is measure this coupling. On the charge sense resistor he makes a voltage step. Then the resistor is going to charge up the person. If you look at this charging curve, there's an origin, a slope and an intercept. You can model the human as a series and a parallel resistor and a capacitor





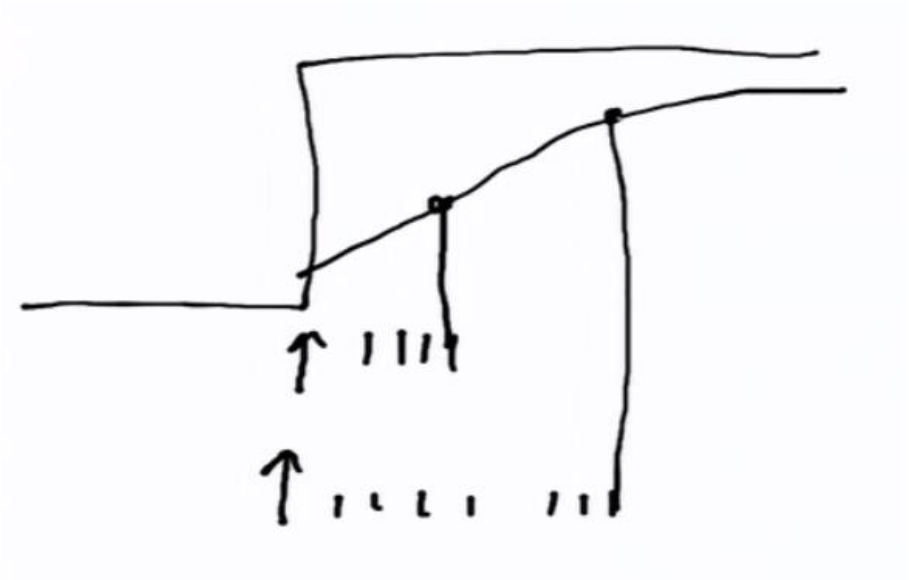
Below drawing of the human as a series and a parallel resistor and a capacitor



Use this as a model for the human and then by looking at this step response. By measuring the origin, slope and intercept, you can separately measure the series in parallel, resistance and capacitance.

There is a key trick here which is if you take this step response and you look at the curve, the time scale of the whole step response is in the order of micro seconds. Its fast. Its faster than the A to D can read. but if you go back to the A to D data sheet, remember that before the A to D is a sample and hold. The sample and hold is an analog memory that stores a reading. So what we'll do is a second trick called sychronist undersampling. So we fire the charging poles. then the processor, we can count some number of processor cycles. Then we initiate the A to D conversion. the sample and hold remember it and then the conversion is slow. then i can go back and can repeat the measurement but change the delay and read it again. Then do that a few different times. So with

multiple measurement we can trace out the charging curve. Below image illustrates this. We can do that because we control the stimplus that drive it.

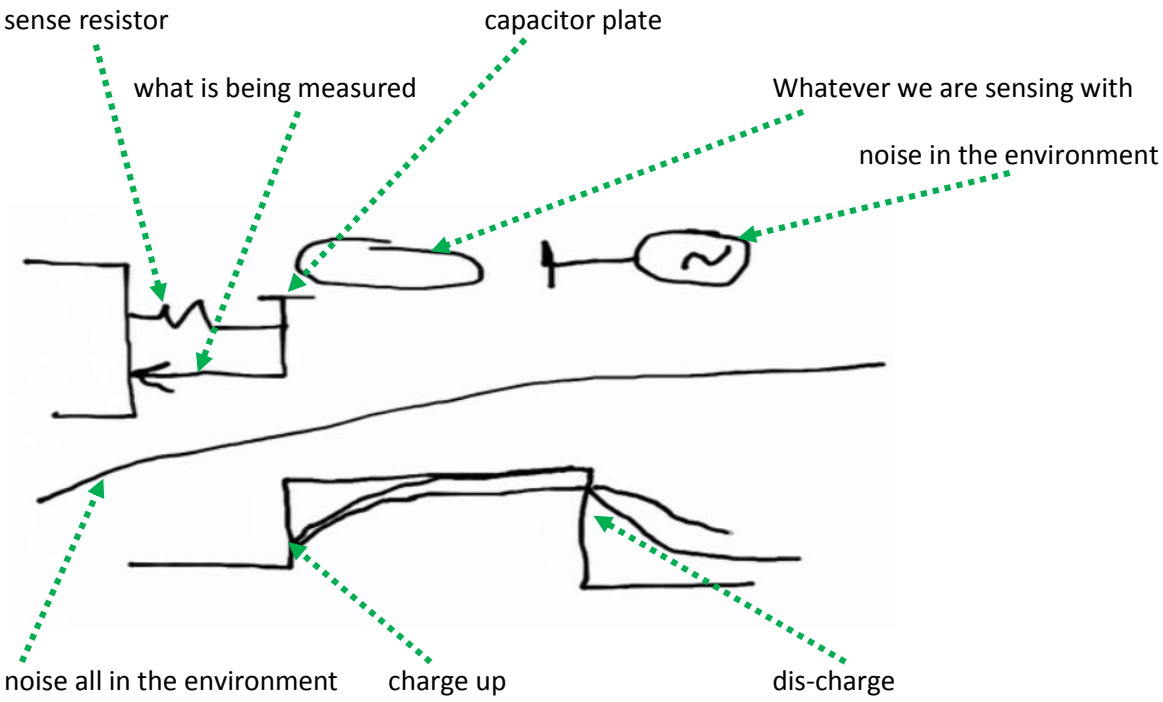


There are three different points on the charging curve. Right now we don't need all of them. we're only measuring proximity but we could use them to do things like materials charcatisation. You can measure impurities in wood or soil or moisture or humidity.

C programme code - in the header Neil defines 3 different delays.

Neil explained one more trick to reduce noise using the software.

He drew this two ways



Worst noise is 60 cycles from power lines or 50 cycle. We got extra noise coming in. If we look at the charge then it's going to charge up. Over all over this is external noise. Depending on where we are in the 60 cycle it might charge faster or slower. But if we charge and then dis-charge. so it will charge up and then down. Depending on where we are in the cycle it might make it charge faster and then dis-charge slower. So the noise is asymmetrical with going up and going down. What Neil does in the software is charge up and read it. Charge down and read it. And if you add those two times they separate to first order noise coming in. It's a software trick for noise reduction.

Going to settle the electrode, charge it, and wait. Settle the electrode, discharge it, and wait and read it. And save that. Send it out do the same thing at a different delay and then do the same thing at a different delay still.

```
  while (1) {
     //
     // send framing
     //
     put_char(&serial_port, serial_pin_out, 1);
     char_delay();
     put_char(&serial_port, serial_pin_out, 2);
     char_delay();
     put_char(&serial_port, serial_pin_out, 3);
     char_delay();
     put_char(&serial_port, serial_pin_out, 4);
     //
     // settle, charge, and wait 1
     //
```

```
      settle_delay();
      set(charge_port, charge_pin);
      charge_delay_1();
      //
      // initiate conversion
      //
      ADCSRA |= (1 << ADSC);
      //
      // wait for completion
      //
      while (ADCSRA & (1 << ADSC))
         ;
      //
      // save result
      //
      up_lo = ADCL;
      up_hi = ADCH;
      //
      // settle, discharge, and wait 1
      //
      settle_delay();
      clear(charge_port, charge_pin);
      charge_delay_1();
      //
      // initiate conversion
      //
      ADCSRA |= (1 << ADSC);
      //
      // wait for completion
      //
      while (ADCSRA & (1 << ADSC))
         ;
      //
      // save result
      //
      down_lo = ADCL;
      down_hi = ADCH;
      //
      // send result
      //
      put_char(&serial_port, serial_pin_out, up_lo);
      char_delay();
      put_char(&serial_port, serial_pin_out, up_hi);
      char_delay();
      put_char(&serial_port, serial_pin_out, down_lo);
      char_delay();
      put_char(&serial_port, serial_pin_out, down_hi);
      char_delay();
      //
      // settle, charge, and wait 2
      //
      settle_delay();
      set(charge_port, charge_pin);
      charge_delay_2();
      //
      // initiate conversion
      //
      ADCSRA |= (1 << ADSC);
      //
```

```
      // wait for completion
      //
      while (ADCSRA & (1 << ADSC))
         ;
      //
      // save result
      //
      up_lo = ADCL;
      up_hi = ADCH;
      //
      // settle, discharge, and wait 2
      //
      settle_delay();
      clear(charge_port, charge_pin);
      charge_delay_2();
      //
      // initiate conversion
      //
      ADCSRA |= (1 << ADSC);
      //
      // wait for completion
      //
      while (ADCSRA & (1 << ADSC))
         ;
      //
      // save result
      //
      down_lo = ADCL;
      down_hi = ADCH;
      //
      // send result
      //
      put_char(&serial_port, serial_pin_out, up_lo);
      char_delay();
      put_char(&serial_port, serial_pin_out, up_hi);
      char_delay();
      put_char(&serial_port, serial_pin_out, down_lo);
      char_delay();
      put_char(&serial_port, serial_pin_out, down_hi);
      char_delay();
      //
      // settle, charge, and wait 3
      //
      settle_delay();
      set(charge_port, charge_pin);
      charge_delay_3();
      //
      // initiate conversion
      //
      ADCSRA |= (1 << ADSC);
      //
      // wait for completion
      //
      while (ADCSRA & (1 << ADSC))
         ;
      //
      // save result
      //
      up_lo = ADCL;
```

```
      up_hi = ADCH;
      //
      // settle, discharge, and wait 3
      //
      settle_delay();
      clear(charge_port, charge_pin);
      charge_delay_3();
      //
      // initiate conversion
      //
      ADCSRA |= (1 << ADSC);
      //
      // wait for completion
      //
      while (ADCSRA & (1 << ADSC))
         ;
      //
      // save result
      //
      down_lo = ADCL;
      down_hi = ADCH;
      //
      // send result
      //
      put_char(&serial_port, serial_pin_out, up_lo);
      char_delay();
      put_char(&serial_port, serial_pin_out, up_hi);
      char_delay();
      put_char(&serial_port, serial_pin_out, down_lo);
      char_delay();
      put_char(&serial_port, serial_pin_out, down_hi);
      char_delay();
      }
```

Then in the software. You get the different readings in and update those bar graphs.

This version will measure anything loading the electrode.