ALGONQUIN COLLEGE

CST8227 INTERFACING

Lecture Week 8

Agenda:

1. Admin.
2. Benefits of Calibration
3. Capacitance sensor
4. Lab #7
5. Piezo buzzer transducer
6. Finish Sensors
7. Finish ADC

Benefits of Calibration: to characterize the output response

1. Sensors may show inconsistent changes in response when subjected to variable conditions such as heat, cold, light.
2. To account for differences in manufacturing - two sensors from the same manufacturer production run may yield slightly different readings.
3. Differences in sensor design mean two different sensors may respond differently in similar conditions.
4. Some sensor technologies 'age' and their response will naturally change over time - requiring periodic re-calibration.
5. To define a way to base the range of outputs being sent to an actuator (an output device), that based on input maximum and minimum values that are obtained from calibration.

Calibration of Sensor

- Signals from transducers are conditioned to interface to a microprocessor such that it fills the complete conversion window of the analog-to-digital (ADC) conversion unit.

  V_{max}: The maximum voltage that an ADC can handle (ex. 3.3 volts)

  V_{min}: The minimum voltage that an ADC can handle (ex. 0 volts)

  $V_{max} – V_{min}$: The conversion window is between these ranges

- Any voltage higher than V_{max} can be coded to default to V_{max}.
- Any voltage lower than V_{min} can be coded to default to V_{min}.
- Tracking “rogue” readings outside of the expected values that are defined with calibration are noteworthy.

Lab 7: Capacitive Touch Sensor

Objectives:

1. Gain familiarity with the touchRead() function to measure capacitance from the microcontroller
2. Gain familiarity with the tone() function to generate sound from the Arduino – see tone.cpp

   https://github.com/PaulStoffregen/cores/blob/master/teensy/Tone.cpp
3. Experiment with a touch sensor as an input device
4. Experiment with a piezo buzzer as an output device

touchRead()

- Only 12 pins on the Teensyduino 3.2 have this functionality.
- It works similarly to analogRead(pin).
- Invoke the function with one of the pins that has touch sensing capability
- Returns a **16** bit number (2^{16} -1 = 65535) .
- This 16-bit number represents the capacitance on that pin, in 1/50th pico-Farad units.
- Ex. If the pin has 40 pF, you'll get 2000.
- Capacitive sensing can also be implemented using ordinary pins with CapSense library.
- Refer to touch.c

  https://github.com/PaulStoffregen/cores/blob/08b835afb8bc4e3adc5b0173b88c20c69abde2a1/teensy3/touch.c

https://blog.adafruit.com/2012/10/19/teensy-3-0-software-update/

Teensy 3.2’s built-in hardware , advantages over software-based capacitive sensing

1. **Sensitivity**: By default, touchRead() gives 0.02 pF sensitivity.
2. **Speed**: The measurement time depends on the capacitance. A worst-case measurement takes about 5 ms. Typical capacitances used for human touch typically read much faster. Even when cycling through all the touch sensitive pins, you get excellent sensitivity at very responsive speeds.
3. **Stability**: The measurement works by comparing the pin's capacitance to an on-chip reference capacitor. If the power supply voltage, charging and discharging currents or other electrical factors change, their effect on the measurement are largely canceled out by using the same on-chip hardware to measure both the pin and the reference capacitor.

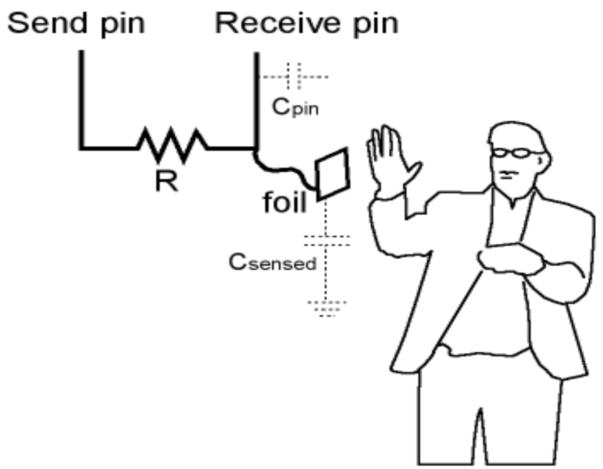
Reference: https://www.kickstarter.com/projects/paulstoffregen/teensy-30-32-bit-arm-cortex-m4-usable-in-arduino-a/posts/331757

Capacitive sensing:
Possible sources of error and troubleshooting

- The capsense functions transmits a 50% duty cycle PWM signal:
  - Digital switching (transition from ‘0’ to ‘1’, etc…) can be “noisy”:
    - presence of large current pulses.
    - The circuitry that the voltages and currents need to pass through.
- Install a capacitor (i.e. 0.1 μF) between the power rails in order to reduce power supply noise.
- The measurement of the capacitance is with respect to ground:
  - Plug your laptop into an electrical source: make certain that Teensy’s ground is connected to earth ground, usually through the USB cable.

CapacitiveSensor Class library

- **CapacitiveSensor** library turns two or more Arduino pins into a capacitive sensor, which can sense the electrical capacitance of the human body.
- Sensor setup requires a resistor (100 kΩ to 50 MΩ) and a piece of wire and a piece of aluminum foil on the end.
- The sensor will start to sense a hand or body inches away from the sensor.
- The capacitiveSensor method toggles a microcontroller *send* pin to a new state and then waits for the *receive* pin to change to the same state as the send pin.
- A variable is incremented inside a *while* loop to time the receive pin's state change.
- The method then reports the variable's value, which is in arbitrary units.
- http://www.youtube.com/watch?v=BHQPqQ_5ulc



tone() (cont'd)

Syntax

- tone(pin, frequency)
  tone(pin, frequency, duration)

Parameters

- pin: the pin on which to generate the tone
- frequency: the frequency of the tone in hertz - *unsigned int*
- duration: the duration of the tone in milliseconds (optional) - *unsigned long*

Returns

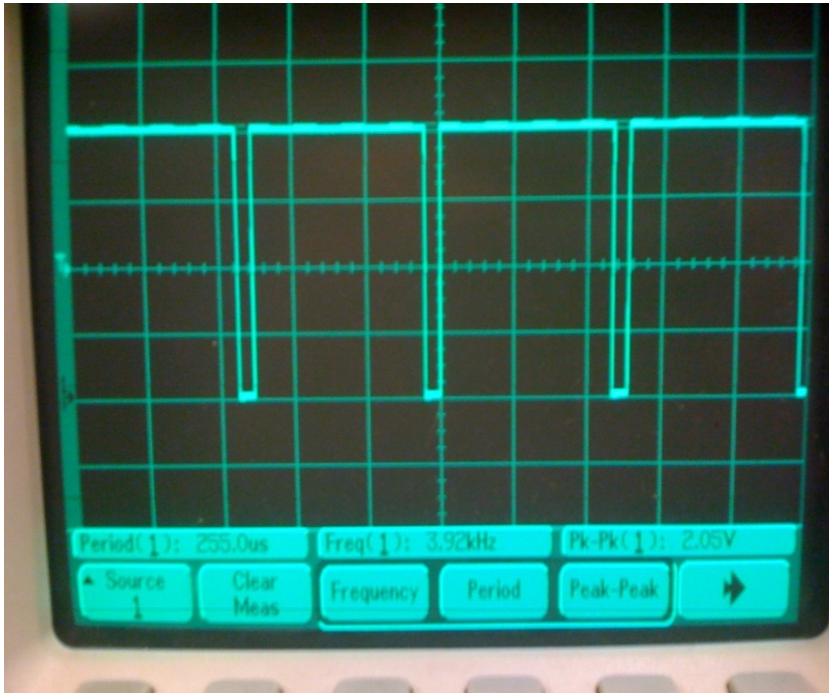
- nothing

tone() Basics

- Generates a square wave of the specified frequency.
- Uses a 50% duty cycle.
- A duration can be specified.
- If no duration is specified the wave continues until a call to noTone().
- Only one tone can be generated at a time.
- If a tone is already playing on a different pin, the call to tone() will have no effect.
- If the tone is playing on the same pin, the call will set its frequency.

- For simulating multiple tones: https://www.arduino.cc/en/Tutorial/BuiltInExamples/toneMultiple
- The tone() command works by taking over one of the processor's internal timers, setting it to the frequency you want, and using the timer to pulse an output pin. Since it's only using one timer, you can only play one note at a time. You can, however, play notes on multiple pins sequentially. To do this, you need to turn the timer off for one pin before moving on to the next.

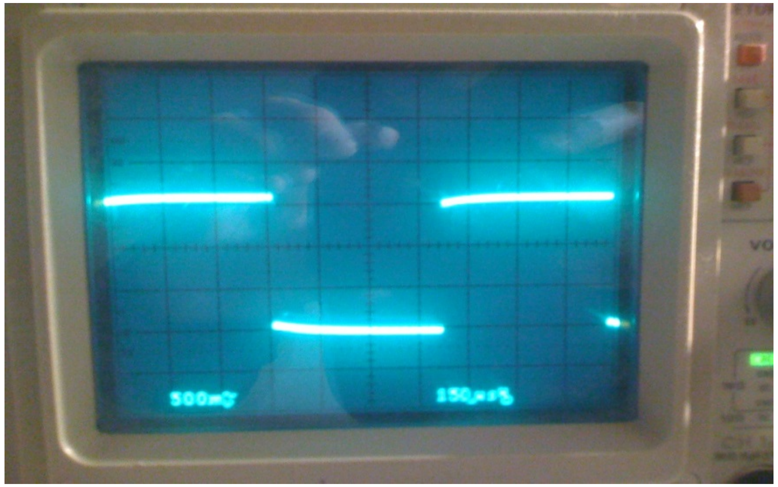
Recall PWM from analogWrite()



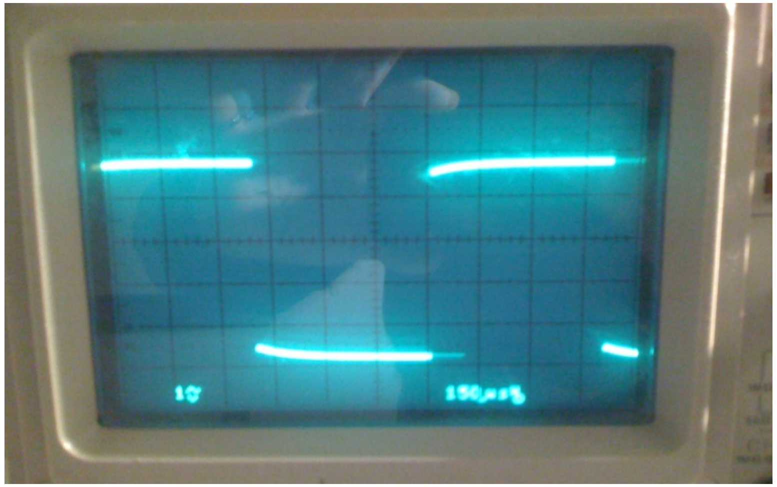
- When you use analogOut() to create pulsewidth modulation (PWM) on an output pin.
- You can change the on-off ratio of the output (also known as the *duty cycle*) but not the frequency.
- If you have a speaker connected to an output pin running analogOut(), you'll get a changing loudness, but a constant tone.
- To change the tone, you need to change the frequency.
- The tone() command does this for you.

http://itp.nyu.edu/physcomp/labs/labs-arduino-digital-and-analog/tone-output-using-an-arduino/

analogWrite() vs. tone(), 976.56 Hz



- Specifications say that Pin 5 puts out a 976.56 Hz PWM
- Calculations f ≈ 980 Hz.
- Duty cycle can be configured



- tone(8, 976);
- Always a 50% duty cycle

Piezo Buzzer Oscilloscope Output from tone()

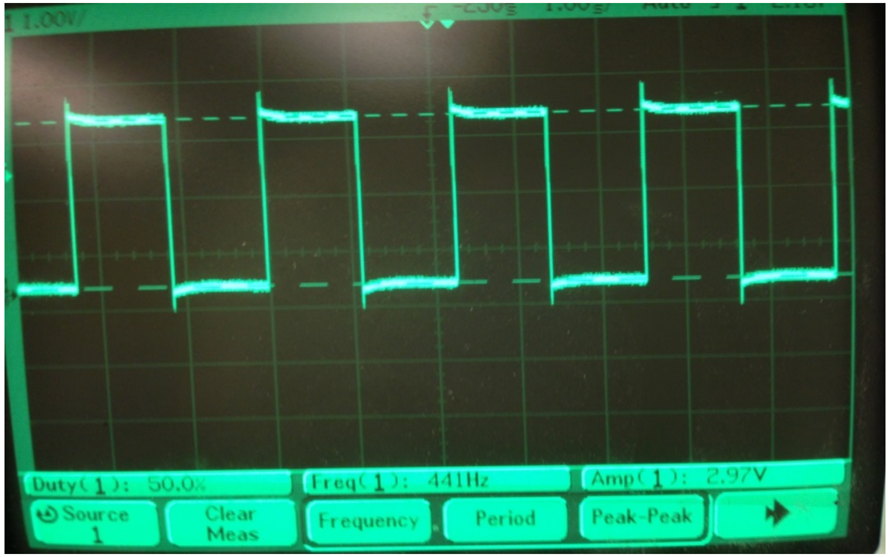
tone(speakerPin, 440);

<u>Expectations:</u>

Duty cycle = 50%

Frequency = 440 Hz

Amplitude ≈ 3.3V



Capacitance

- Capacitance can be defined as the ***ability to store electrical charge***
- It can be very useful:
  - To build tuned filters, oscillators, etc.. in communication circuits.
  - To correct for power factor in power distribution applications.
  - To smooth the “ripple” that occurs when converting from AC to DC signals, etc…..
- Or sometimes unwanted:
  - Coupling between parallel conductors creates fields where power is stored (and not getting to a load) and can also disrupt information carrying signals where the voltage level of the signals is typically very small.

Physical Construction of Capacitors

- A basic capacitor is two conductors separated by insulator.
- Remember the "fruit-rollups" analogy?
- Many capacitance calculations involve integrals and derivatives.
- For a parallel plate capacitor:

$$C = \epsilon_r \epsilon_o \frac{A}{4\pi d} \text{ [Farads]}$$

- ε_r = dielectric constant
- ε_o = electric constant
- A = area of plates
- D = separation of plates

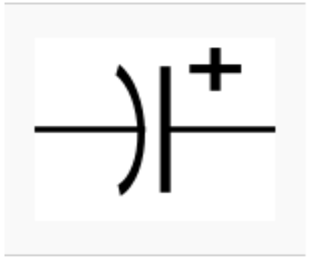
Conductors	Insulators
Copper (Cu)	Wood
Ag, Au, Al*	Glass
Carbon	Rubber
Water….	Dry Air

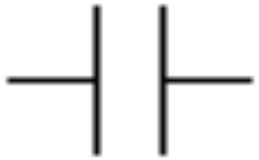
Human body is on the ***conductive*** side of the spectrum

*Refer to periodic table

Images

Schematics





Photographs

Commercial Capacitor Markings

- If units are not specified on the casing of the capacitor, units are likely picoFarads [pF].
- Conventions are similar to resistors in that the first two digits represent significant digits of the value, and the third is a multiplier.

ex.

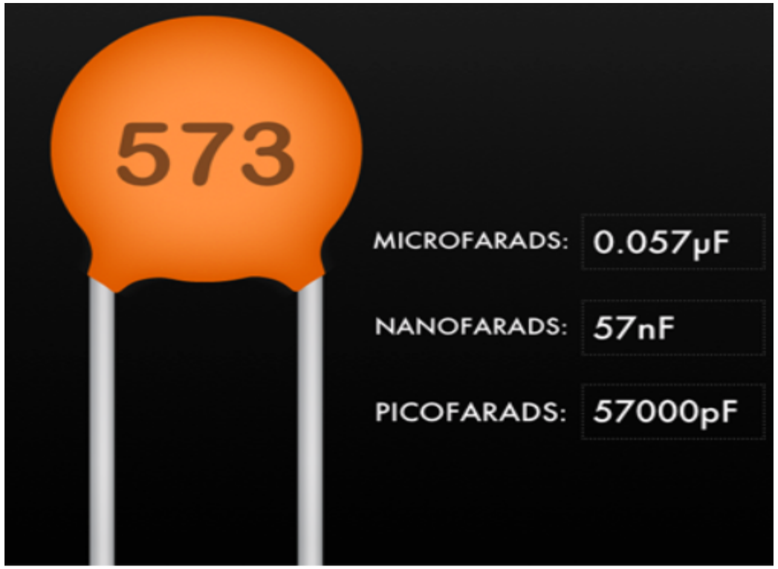
$102 = 10 \times 10^2$ pF

$= 1000$ pF

$= 1000 \times 10^{-12}$F

$= 0.001 \times 10^{-6}$F

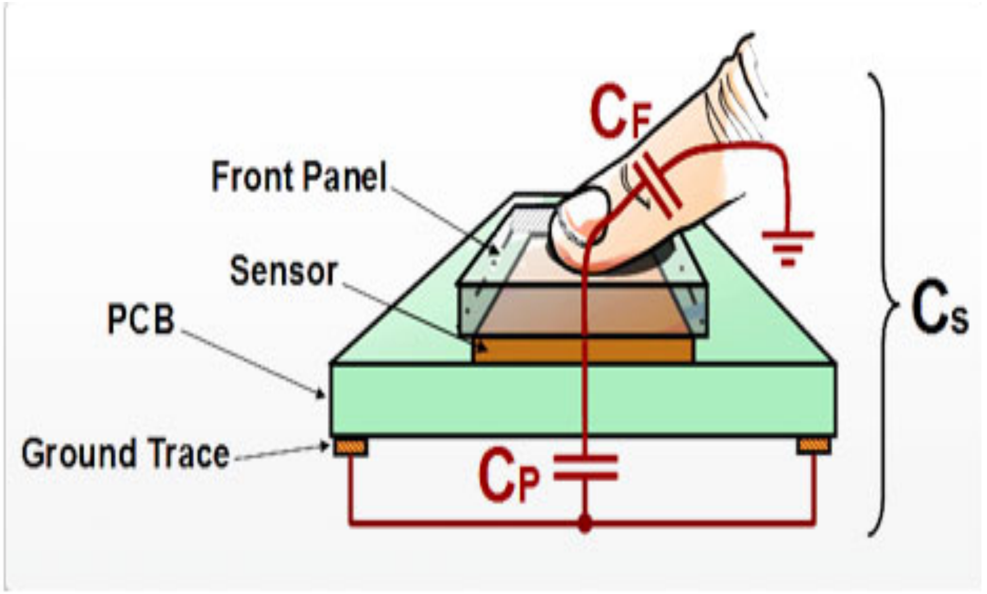
$= 0.001\ \mu$F



Graphic: http://makezine.com/review/toolbox-review-circuit-sidekick/

Human Body and Capacitance

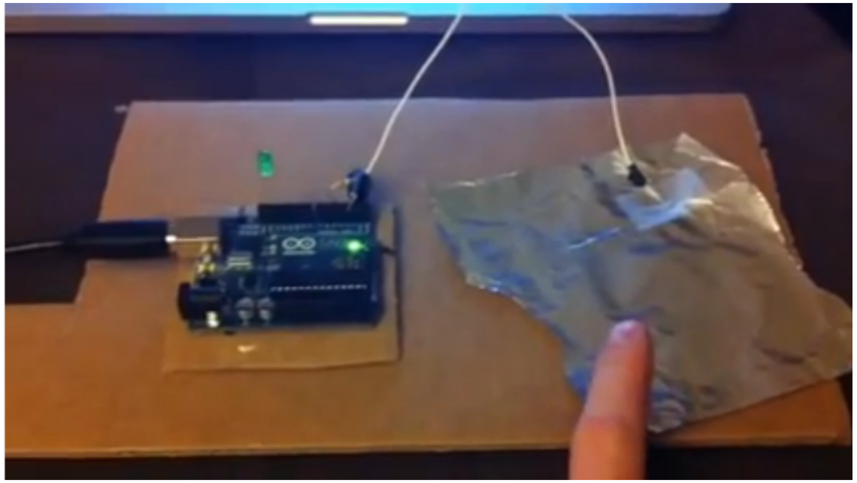
- The Human Body Model for capacitance, as defined by the Electrostatic Discharge Association (ESDA), consists of a 100pF capacitor in series with a 1.5 kΩ resistor (reference: https://www.esda.org/esd-overview/esd-fundamentals/part-5-device-sensitivity-and-testing/).
- Human touch increases capacitance.
- Useful applications include proximity/displacement sensor, humidity sensors, fluid level sensors, HIDs such as touchscreens, etc.….



Natural Capacitance of the Human body and Touch Sensors

- A touch sensor exploits the natural capacitance that a human can present to an electronic circuit.
- A capacitor physically is two conductors separated by an insulator.
- A human becomes one conductor, air is the insulator, and the circuit would be the second conductor.

Capacitive Sensing in Lab #7



- Capacitive sensors fall under the category of ***proximity*** sensors.
- Human body is used as an input.
- Electronic charge is transferred to the foil, either by direct contact or by inducing charge.
- When the input is removed, the electrons discharge through the resistor.

Lab #7: "Time" and Touch Sensor Demo Videos

Calibration – essential for adjusting for differing environmental conditions

1. Sampling the sensor readings for a suitable period of time (eg. 1/8 sec)
2. Binning the results to determine which note is played

Without calibration, no control over timing and pitch of sound produced in response to touch sensor readings

- The physical construction of the touch sensor is also demonstrated:
  - foldable cardboard support for tinfoil
  - tinfoil layer, taped to cardboard
  - paperclip to connect sensing wire to tinfoil

- ... You will *definitely* need to read all the documentation identified in Lab 7 in order to get successful readings from the touch sensor.

Lab #7 Auto-Calibration Required for Sensors

Objective

To account for losses due to capacitance or light couplings in the surrounding environment.

- auto-calibration can be done by simply having an initial period of time (eg. 10-30 secs) where the capacitive touch sensor is being continuously read.
- All values read are checked to determine the minimum and maximum.
- At the end of the time, take the min & max values and use them as "0" level (or Off) and "100%" level for all subsequent readings.

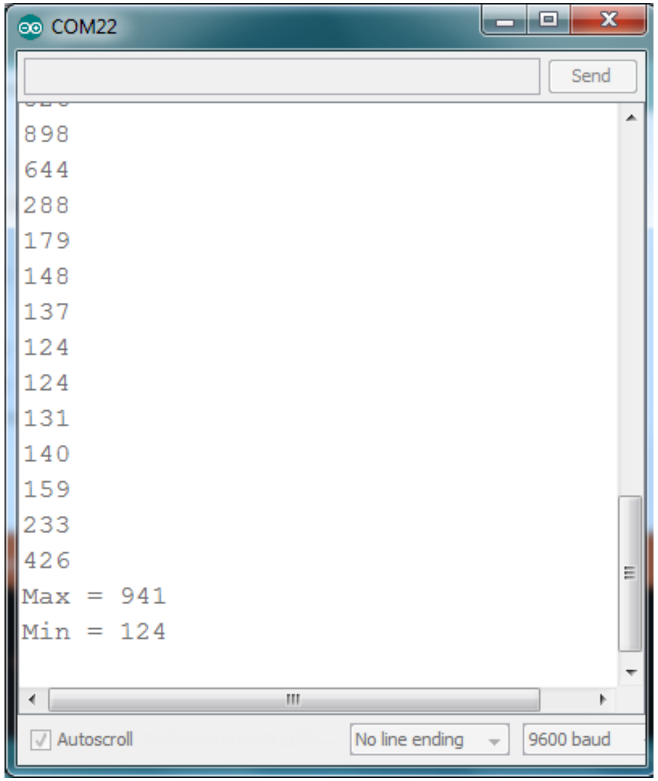
Summary of Results of Light Sensor Calibration

```
void calibrate(){

  for(int i = 0; i < 100; i++)
  {
    reading = analogRead(sensor);

    if(reading < minReading)
        minReading = reading;

   if(reading > maxReading)
         maxReading = reading;
  }
}  //end calibrate
```

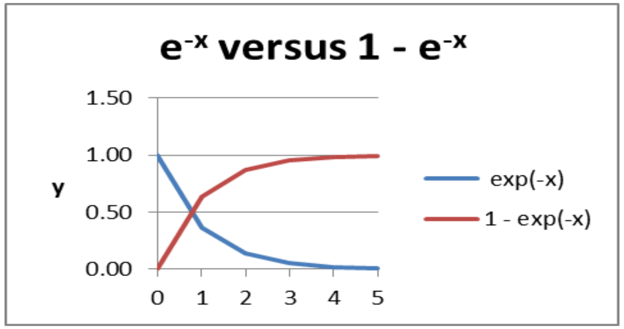


```
898
644
288
179
148
137
124
124
131
140
159
233
426
Max = 941
Min = 124
```

Resistor and Capacitor Choices for Calibration

	Resistor	Results
Resistor	1 MΩ (or less)	Sensor requires absolute touch to activate.
	10 MΩ	sensor will start to respond 4-6 inches away.
	40 MΩ	sensor will start to respond 12-14 inches away.
Capacitor	1 µF	Short discharge time (of electrons)
	10 µF	Longer discharge time (of electrons)

Voltage across a capacitor = $V_C = V^*(1- e^{-\tau/RC})$



map() function - what you will do with the autocalibration data

- The map() function can be used to convert readings into a percentage of full range (of volume or tone) as opposed to doing the math in the code.
- Re-maps a number from one range to another.
- That is, a value of fromLow would get mapped to toLow, a value offromHigh to toHigh, values in-between to values in-between, etc.
- Does not constrain values to within the range, because out-of-range values are sometimes intended and useful. The constrain() function may be used either before or after this function, if limits to the ranges are desired.
- The map() function uses integer math so will not generate fractions, when the math might indicate that it should do so. Fractional remainders are truncated, and are not rounded or averaged.

map() function (cont'd)

Syntax:

map(value, fromLow, fromHigh, toLow, toHigh)

Ex. y = map(x, 1, 50, 50, 1);

Ex.

```
void loop()
{
  int val = analogRead(0);
  val = map(val, 0, 1023, 0, 255);
  analogWrite(9, val);
}
```

Reference: https://www.arduino.cc/en/Tutorial/BuiltInExamples/tonePitchFollower

Protoboard and GND

- Grounding can be an issue, especially for higher frequency or digital circuits.
- Make certain that the ground plate of your protoboard is installed.
- GND provides a return path for currents.
- PCBs have traces that can have higher currents travelling through them: currents -> magnetic field -> coupling of adjacent conductors
- The protoboard's GND plate can aid with:
  1. the reduction of electrical noise.
  2. The reduction of the inadvertently coupling of signals (crosstalk)

Links discussed in class

- https://forum.arduino.cc/index.php?topic=323392.0
- https://play.google.com/store/apps/details?id=com.fennel.ledcontrol
- https://github.com/PaulStoffregen/cores/blob/master/teensy3/touch.c
- https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-touch.h#L43