Johnson Noise

Michael Mulhearn

May 19, 2018

1 Introduction

Johnson–Nyquist noise is the electronic noise that results from the thermal excitation of electrons in a conductor independent of any applied voltage. As the applied voltage is zero, there is no net potential difference due to Johnson noise ($\langle V \rangle = 0$), but generally the average power $\langle V^2 \rangle$ is non-zero. As we will show, the one-sided power spectral distribution turns out to be simply:

$$\mathcal{S}_+(f) = 4Rk_\mathrm{b}T, \tag{1}$$

By measuring all of the other quantities, we will use this relation to experimentally determine Boltzman's constant, which has the value

$$k_\mathrm{b} = 1.38064852(79) \times 10^{-23}\mathrm{J/K}$$

with the uncertainty in parenthesis.

2 Autocorrelation and Power Spectrum Distribution

The signal $V(t)$ associated with random noise does not have a Fourier Transform, and so the obvious mechanism for finding the frequency spectrum associated with $V(t)$ is ruled out. Instead, we define the autocorrelation by:

$$\mathcal{R}(\tau) \equiv \lim_{T\to\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} dt\; V(t)V(t-\tau) = \langle V(t)V(t-\tau)\rangle \tag{2}$$

which is part of a Fourier transform pair:

$$\mathcal{S}(f) \equiv \int_{-\infty}^{\infty} d\tau\; \mathcal{R}(\tau)\exp(-i2\pi f\tau) \tag{3}$$

$$\mathcal{R}(\tau) = \int_{-\infty}^{\infty} df\; \mathcal{S}(f)\exp(i2\pi f\tau) \tag{4}$$

To interpret $\mathcal{S}(f)$ we note that Definition 2 implies that:

$$R(0) = \langle V^2(t)\rangle \equiv P_\mathrm{avg}$$

but from Equation 4 we also have:

$$\begin{aligned} R(0) &= \int_{-\infty}^{\infty} df\; \mathcal{S}(f)\exp(i2\pi f0) \\ &= \int_{-\infty}^{\infty} df\; \mathcal{S}(f) \end{aligned}$$

or in other words:

$$P_{\text{avg}} \equiv \langle V^2(t) \rangle = \int_{-\infty}^{\infty} df\ \mathcal{S}(f)$$

That is to say $\mathcal{S}(f)$ is the average power contained at frequency f. To calculate $\mathcal{S}(f)$ we simply calculate the Fourier transform of the autocorrelation function:

$$\mathcal{R}(\tau) \equiv \langle V(t)V(t-\tau) \rangle . \tag{5}$$

There are a few practical simplifications we can make resulting from the fact that $\mathcal{R}(\tau)$ is a real even function, and therefore need only calculate:

$$\mathcal{S}(f) = 2\int_0^{\infty} d\tau\ \mathcal{R}(\tau)\cos(2\pi f\tau).$$

But now clearly $\mathcal{S}(f)$ is also an even function, and we can therefore consider the one-sided-power spectral distribution:

$$\begin{aligned} \mathcal{S}_+(f) &\equiv 2\mathcal{S}(f) & (6)\\ &= 4\int_0^{\infty} d\tau\ \mathcal{R}(\tau)\cos(2\pi f\tau) & (7) \end{aligned}$$

which has the same interpretation as the power spectral distribution but simply restricted to positive frequencies:

$$P_{\text{avg}} \equiv \langle V^2(t) \rangle = \int_0^{\infty} df\ \mathcal{S}_+(f) \tag{8}$$

3 Theory of Johnson Noise

Johnson–Nyquist noise is the electronic noise that results from the thermal excitation of electrons in a conductor independent of any applied voltage. As the applied voltage is zero, there is no net potential difference due to Johnson noise ($\langle V \rangle = 0$), but generally the average power $\langle V^2 \rangle$ is non-zero. As we will show, the one-sided power spectral distribution turns out to be simply:

$$\mathcal{S}_+(f) = 4RkT, \tag{9}$$

where the one-sided power spectral distribution $\mathcal{S}_+(f)$ is the average power per bandwidth, i.e.

$$P_{\text{avg}} \equiv \langle V^2(t) \rangle = \int_0^{\infty} df\ \mathcal{S}_+(f)$$

Consider a resistor of resistance R with no potential imposed across it. Using the superposition principle, we'll consider that the total potential V across the resistor then results from sum of the voltage caused by each individual electron due to it's current I_i:

$$V_i(t) = RI_i = \frac{Re}{L}u_i(t)$$

where L is length of the resistor and u_i is the velocity along the axis of the resistor. With no applied voltage and therefore no preferred direction we must have $\langle u_i \rangle = 0$. However, because each electron is in thermal equilibrium we have $m\langle u_i^2 \rangle = kT$ and so:

$$\langle V_i^2(t) \rangle = R^2\frac{e^2}{mL^2}kT.$$

Assuming each electron is uncorrelated with the other electrons,

$$\langle V_i(t)V_j(t)\rangle = 0,$$

for $i \neq j$ and so:

$$\langle V^2(t)\rangle = \sum_{ij} \langle V_i(t)V_j(t)\rangle = \sum_i \langle V_i^2(t)\rangle = R^2 \frac{Ne^2}{mL^2} kT,$$

where N is the number of electrons.

The Fourier transform of $V(t)$ is not defined for continuous random noise. However, we can still determine the power spectral distribution from the autocorrelation function, as discussed above. In this case, the electrons are subject to collisions which alter their trajectory on a timescale $\tau_{\rm c}$, and we therefore expect the signal to become uncorrelated on that timescale:

$$\mathcal{R}(\tau) \equiv \langle V(t)V(t-\tau)\rangle = A\exp(-\tau/\tau_{\rm c})$$

for $\tau \geq 0$, which is all we will need. We note that by definition

$$\mathcal{R}(0) = \langle V^2(t)\rangle$$

and so we must have:

$$\mathcal{R}(\tau) = R^2 \frac{Ne^2}{mL^2} kT \exp(-\tau/\tau_{\rm c}).$$

With the autocorrelation function in hand, the one-sided power spectrum distribution is only an integral away:

$$\begin{aligned}
\mathcal{S}_+(f) &= 4R^2 \frac{Ne^2}{mL^2} kT \int_0^{-\infty} \cos(2\pi f\tau)\exp(-\tau/\tau_{\rm c}) \\
&= 4R^2 \frac{Ne^2}{mL^2} kT \frac{\tau_{\rm c}}{1+(2\pi f\tau_c)^2} \\
&= 4RkT \frac{1}{1+(2\pi f\tau_c)^2}
\end{aligned}$$

Where in the last step we have used the fact the resistance is given by:

$$R = \frac{mL^2}{Ne^2\tau_c}.$$

Noting that $\tau_c \sim 10^{-14}$ s for typical conductors, then at any frequency we are likely to encounter in an electric circuit $f\tau_c << 1$ so that the one-sided power spectral distribution is simply

$$\mathcal{S}_+(f) = 4RkT. \tag{10}$$

4 Experimental Expectations

We saw in the previous sections that the one-sided power spectral distribution is defined by

$$P_{\rm avg} \equiv \langle V^2(t)\rangle = \int_0^\infty df\ \mathcal{S}_+(f) \tag{11}$$

The average power is something we can easily measure. When we measure the RMS of a signal we are measuring

$$V_{\rm rms} = \sqrt{\langle V^2(t)\rangle} = \sqrt{P_{\rm avg}}$$

To make a quantitative prediction for the result of this measurement, we integrate the power spectral distribution across the entire bandwidth for the circuit (all electronic circuits cut off at some high-frequency).

In the case of Johnson noise we found that:

$$\mathcal{S}_+(f) = 4Rk_{\rm b}T. \tag{12}$$

which is flat in frequency space. Note that at room temperature:

$$4k_{\rm b}T = 1.6 \times 10^{-5} \frac{\rm mV^2}{\rm kHz M\Omega}$$

so even with a 1 MΩ and a 10 kHz bandwidth, the RMS voltage due to Johnson Noise will be much smaller than 1 mV. We will need to amplify this signal with very high gain in order to see the effect of Johnson Noise. Our gain will have a frequency dependence, even though our power spectral distribution does not, so we have:

$$\begin{aligned} V_{\rm rms}^2 &= \int_0^\infty df\ g^2(f)\mathcal{S}_+(f) \\ &= 4k_{\rm b}TR\int_0^\infty df\ g^2(f) \end{aligned}$$

If we approximate our circuit as having a constant gain G across a bandwidth Δf, we have:

$$V_{\rm rms}^2 = 4k_{\rm b}TRG^2\Delta f$$

The high gain devices we will be using have a bandwidth of about 10 kHz and a fairly constant gain in the range $3000 - 5000$. The devices allow you to select between resistors with values $R = 1\ \text{M}\Omega, 500\ \text{k}\Omega, 100\ \text{k}\Omega, 60\ \text{k}\Omega, 30\ \text{k}\Omega$.

For a gain in the range $G = 3500$ to 4000 and $R = 1$ MΩ this would yield a power spectral density in the range of:

$$200 - 262\ \text{mV}^2/\text{kHz}$$

For a 10 kHz bandwidth, the corresponding expected RMS voltage would be $44 - 51$ mV.

In the first week of this experiment, you will measure the gain of the Johnson Noise circuit as a function of frequency using the attenuated output from your function generator. You will then measure the RMS voltage resulting from amplified Johnson noise and use this to determine $k_{\rm b}$.

In the second week of this experiment, you will build a power spectrum analyzer, which will allow you to qualitatively investigate the (flat) Johnson noise spectrum, as well as extract $k_{\rm b}$ more directly from the average PSD.

5 Johnson Noise Device Gain Measurement

In this section, we will measure the gain of the Johnson Noise device as function of frequency.

Obtain a Johnson Noise (JN) measurement device (Model JN-A or JN-B) and note the serial number (e.g. #4). These are high gain devices that are very susceptible to damage. Do not drive them directly from a function generator, but use a 1000:1 attenuator, taking care that the small

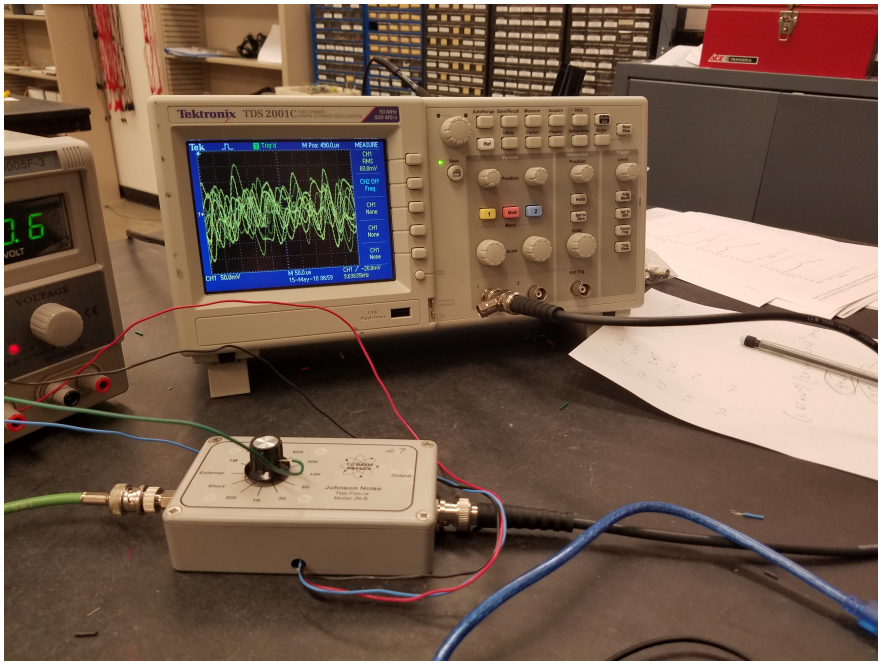


Figure 1: Connections to the Johnson Noise device. Note the grounding wire used to reduce noise from the knob acting as an antenna!

resistor is toward the JN device (else you will reduce voltage by a factor 999/1000, and likely damage the equipment!)

The JN devices are very sensitive, easily pickup noise, and ring at a characteristic frequency of about 27 kHz. The tuning knob in particular picks up noise. An old version banana plug cable plugged into the knob at the set screw grounds the knob, eliminated this source of noise. The effect can be quite dramatic.

Adjust the supply levels on your bench-top supply to +12 V and −12 V. Then, with the supply turned off, connect the JN device to power and ground, as labeled on the wires. Once connected, power the device on and adjust the current thresholds to just slightly above where the supplies become current limited. Under normal operation, the JN device should draw about 20 mA from each supply... keep an eye on it. If one of the supplies is inadvertently disconnected, the circuit latches in a high current state, which should be avoided if possible.

Set your function generator to produce a sine wave with $V_{\rm rms} = 1$ V and $f = 1$ kHz. Connect the function generator to channel 1 of your scope, then to the 1000 : 1 attenuator, and then to your JN device. Make certain that the JN device is set to the "External" mode. The output of the JN device should then be plugged directly into the scope channel 2. This configuration puts significantly larger voltage at the JN device than a typical Johnson Noise signal. Fortunately, the devices are quite linear, and this additional voltage will help overcome some sources of noise.

Make a series of gain measurements at different frequencies that will allow to determine the integral

$$\int_{-\infty}^{\infty} df g(f)^2.$$

While making the gain measurement, you will see that the output of the Johnson Noise seems to bounce around. The effect can be seen if Fig. 2. There is a significant amount of low frequency noise in the output. I found the best approach was to trigger on the AC line, which allows you to find a region of the plot with relatively little 60 Hz noise. Use the scopes Measure capability to find

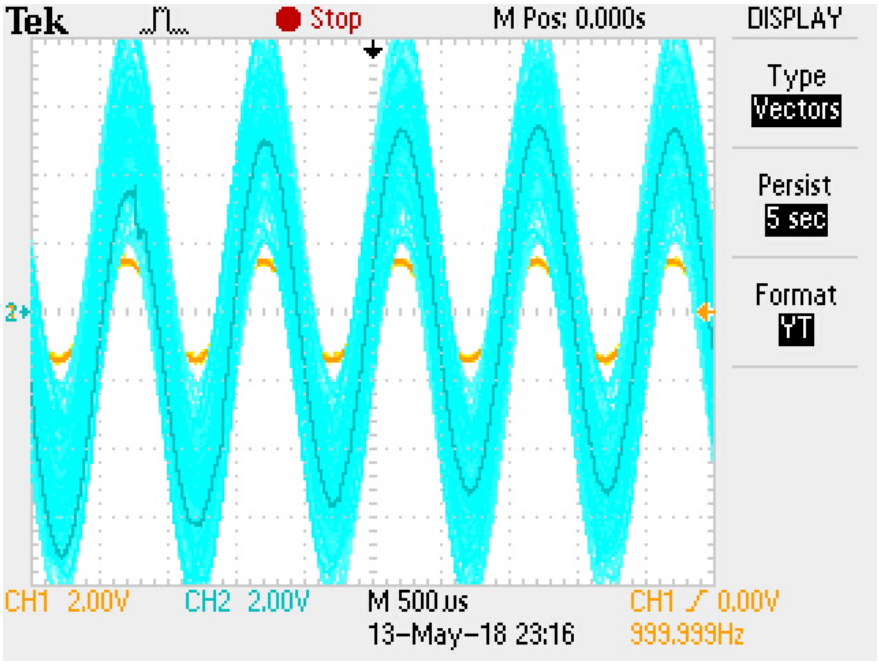


Figure 2: The output of the Johnson Noise device bounces around a lot due to low frequency noise. Here the output is shown with display persistance.

the RMS voltage of the input and output of the Johnson Noise. This is likely the major systematic uncertainty in your measurement, so spend some time thinking about how to quantify it.

You might be tempted to try and use your DMM to measure the RMS voltage of your signal, but this is a mistake. Your DMM is designed to measure 60 Hz AC.

6 Johnson Noise Measurement

You are now ready to make your first-pass Johnson Noise measurement:

- Turn down the function generator and disconnect it from the JN device.
- Turn the knob to select a resistor, and note the RMS voltage from the scope. Repeat this measurement for every resistance.

These measurements, combined with the gain measurement, should allow you to calculate k_b. Before leaving, do make certain that the data you have collected is reasonable. At this stage, within an order of magnitude of k_b is acceptable. **This is the end point for week one.**

7 Digital Dynamic Range Adjustment

In the second week of the lab, you will build a digital spectrum analyzer based on the Arduino Uno. The Arduino's analog inputs have an input voltage range from 0 to 5 V, but our signals are ground referenced (symmetric about 0 V). A DC level shifter is therefore needed to map the ground referenced output of the Johnson Noise device to the 0 to 5 V range expected by the Arduino analog inputs.

Of course we'll be running the ADC as fast as possible which comes at the price of reduced resolution to 8 bits. That would result in a voltage precision of $5\ \mathrm{V}/2^8 \sim 20$ mV. Considering we

will be looking at ~ 100 mV signals, this is just not enough precision. Fortunately, we have one last trick up our sleeve! The ADC has an (optional) external voltage reference, which sets the scale used by the ADC. Experimentation showed that the ADC works reliably down to a voltage reference of ~ 780 mV, which, at 8-bit precision, would give us a 3 mv voltage precision, plenty good enough!

The low-pass filter in Fig. 3 is used to convert PWM output (on pin 5 of the Arduino) into a DC reference voltage V_{ref}. This reference voltage is used as (1) the reference for the ADC, at pin AREF, setting the dynamic range of the ADC, and (2) the reference to the DC level shifter circuit in Fig. 4 so that input to the ADC is referenced to $V_{\text{ref}}/2$. In this way, we can adjust V_{ref} to match the dynamic range of our signal, and the DC offset will be set to the middle of this range. This will ensure that even for small signals, the full 8-bit precision of the ADC is being put to use.

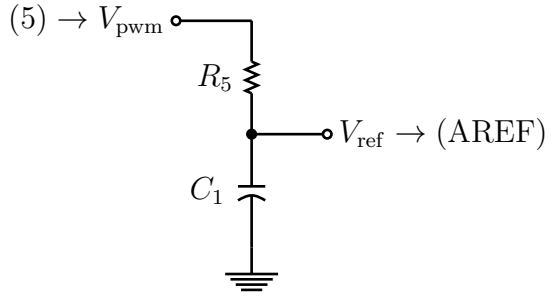


Figure 3: A low pass filter used to convert the Arduino PWM output (on pin 5) to an analog DC reference voltage (sent to pin AREF and also used by the DC level shifter.)

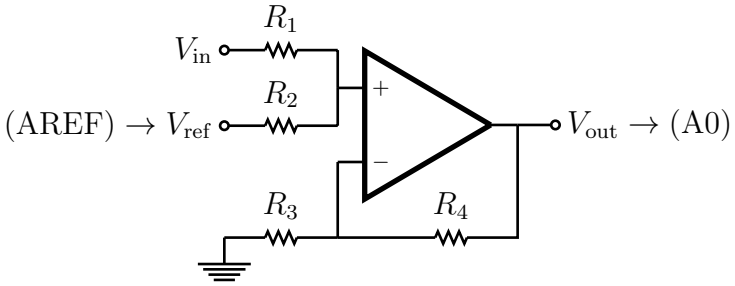


Figure 4: A DC level-shifter with unit gain. The points (AREF) and (A0) refer to the Arduino pins, which should be connected as shown.

For our op-amp we will use a LM358 low-power dual op-amp IC with pinout in Fig. 5. The LM358 is designed to run off a single supply down to 5 V, so we can power this circuit entirely from the Arduino. This is not just convenient, it also ensures that it's output cannot exceed the 0 to 5 V range expected by the Arduino.

You will build the circuits in Fig. 3 and 4 using $R_1 = R_4 = 33$ kΩ, $R_2 = R_3 = 15$ kΩ, $R_5 = 1$ $k\Omega$. For C_1 use a large polarized capacitor ($C_1 = 330$ μF) and **make certain that the negative terminal is connected (as shown in the circuit) to ground.** Connect the positive supply of the Op Amp to the 5 V supply of the Arduino and the 0 V supply to the Arduino ground.

A light-weight sketch CircuitTester is provided which is useful while developing your circuit. This sketch implements an RMS voltmeter (significantly better, if less durable, than the one provided by your DMM.) based on the circuit your are building. It provides control of the PWM on pin 5, reads the analog input on A0, and regularly reports the RMS voltage (and other useful quantities)

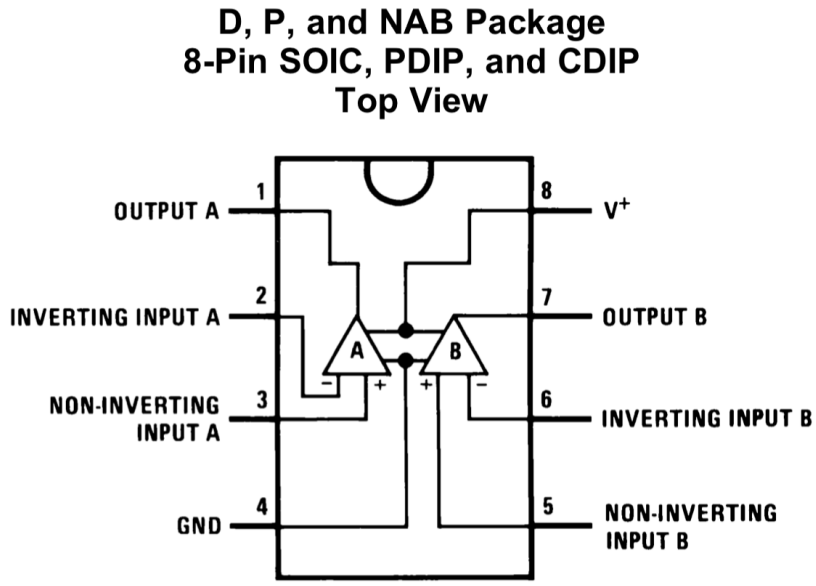


Figure 5: Pinout for the LM358 dual-op-amp with single-ended supply. We'll only need one of the op-amps.

on the serial output. The level of the PWM is specified in the parameter RUN_SCALE. A value of 0 results in 0% duty-cycle for the PWM, a value of 255 corresponds to 100% duty-cycle.

While testing your circuit, you should use your function generator to drive the input V_{in} directly. A 5 kHz sine function with an RMS amplitude of 100 mV is a good starting point. Make sure the ground of the function generator is connected to the ground of your Arduino.

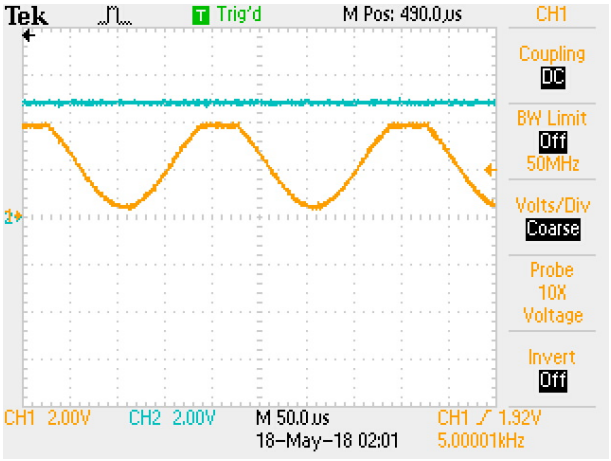
By now we've learned to divide and conquer technical challenges! I'd propose something like:

- Leave the Arduino unpowered.
- Build the level shifter as shown but at first (1) set the reference voltage to 5 V and (2) connect the output to channel one on your scope instead of pin A0.
- Power up the Arduino.
- On your scope you should see the function generator referenced to approximately 2.5 V.
- Power down the Arduino.
- Build the lowpass filter as shown, but connect V_{ref} to channel two on our scope instead of to pin `AREF`.
- Power up the Arduino and download the CircuitTester sketch, setting the parameter `RUN_SCALE` to 112.
- On your scope, you should observe V_{ref}, it should be at about 2.5 V with very little ripple (use AC line trigger and DC offset).
- Adjust the `RUN_SCALE` parameter and ensure you have the expected behavior.
- Power down the Arduino.

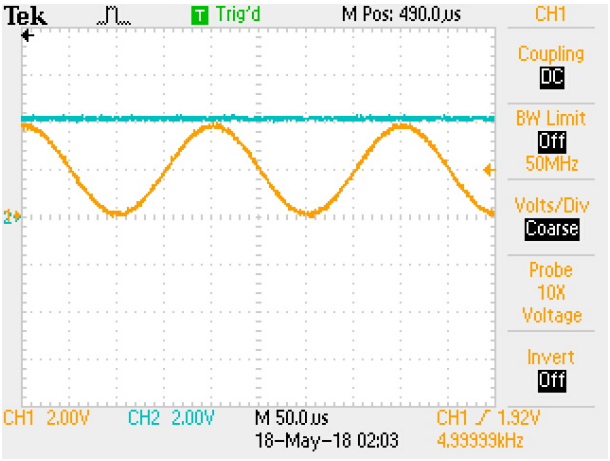
- Connect the lowpass filter output V_{ref} to both (AREF) and the DC level-shifter input V_{ref}.
- Power up the Arduino.
- Check that level-shifter output has the correct DC level when you adjust `RUN_SCALE`.
- Connect the output of the DC level-shifter to the Arduino analog input A0.
- Start the Serial Monitor.

Once the whole circuit is built, the Arduino is running the `CircuitTester` sketch, and the Serial Monitor is open, you should observe periodic RMS voltage readings that should correspond approximately with the output of the function generator.

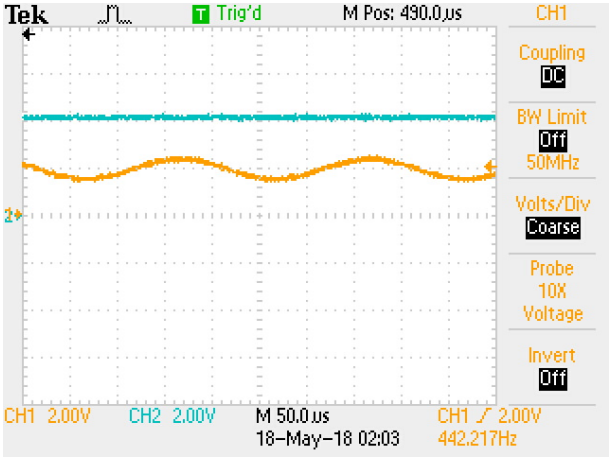
8 Calibration of Arduino Circuit



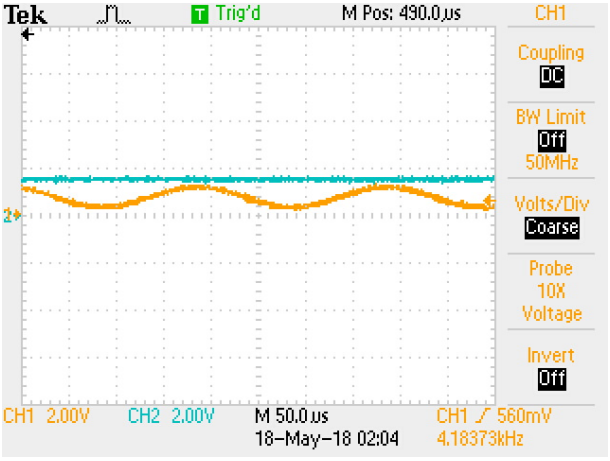
(a) PWM=255, $V_{\text{rms}} = 1.3$ V



(b) PWM=220, $V_{\text{rms}} = 1.3$ V



(c) PWM=220, $V_{\text{rms}} = 300$ mV



(d) PWM=80, $V_{\text{rms}} = 300$ V

Figure 6: Output of DC level-shifter compared to V_{ref} for various settings of the PWM analog output level and the function generator input RMS voltage.

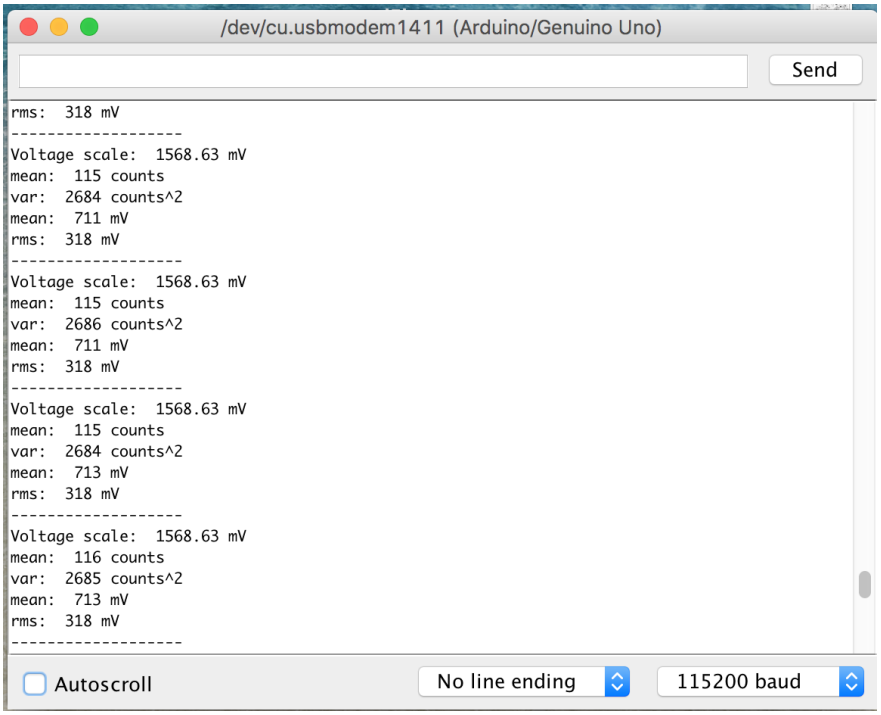
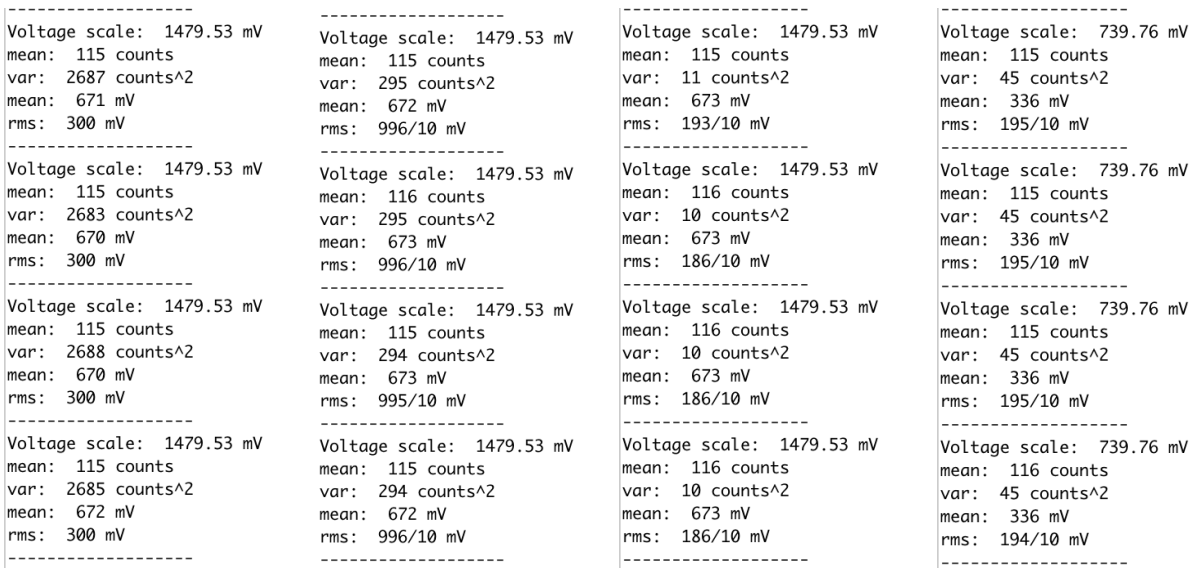


Figure 7: Serial Monitor output for an input signal with $V_{\text{rms}} = 300$ mV. It is systematically off at 318 mV due to full scale being different than nominal 5 V. The calibration is made by changing the value of `CALIB_MV` from 5000 to something more appropriate. The mean, in RAW counts, is right where it should be: near 255/2.



(a) (b) (c) (d)

Figure 8: Post-calibration output for an input signal with (a) $V_{\text{rms}} = 300$ mV (b) $V_{\text{rms}} = 100$ mV (c) $V_{\text{rms}} = 20$ mV (d) $V_{\text{rms}} = 20$ mV. In (a)-(c), the PWM analog output level is set to 80, a good choice for the gain calibration. In (d) the range has been reduced to 40, resulting in better precision. Also note that, for example, 186/10 mV should be read as 18.6 mV... it seems `sprintf` with float is not supported by the Arduino IDE.

```
connecting to the Arduino...
Arduino Initialized
Press Enter to Acquire...
1
1
sending acquire command:  a 1 1
receiving payload from Arduion of length  1538  x  1

voltage scale is  80
run  0
x:  0 y:  124
x:  1 y:  114
x:  2 y:  104
x:  3 y:  97
x:  4 y:  92
sample rms is  99.6950387241
peak at f= 5.0  with Vrms= 99.5284272594
100
saturated count is  0
total integrated rms is  99.6950387241
in narrow region of peak is  99.6515233356
in wide region of peak is  99.6598663342
mean of PSD:   258.15846094 mV^2/kHz
mean of PSD in region:   1985.58230966 mV^2/kHz
```

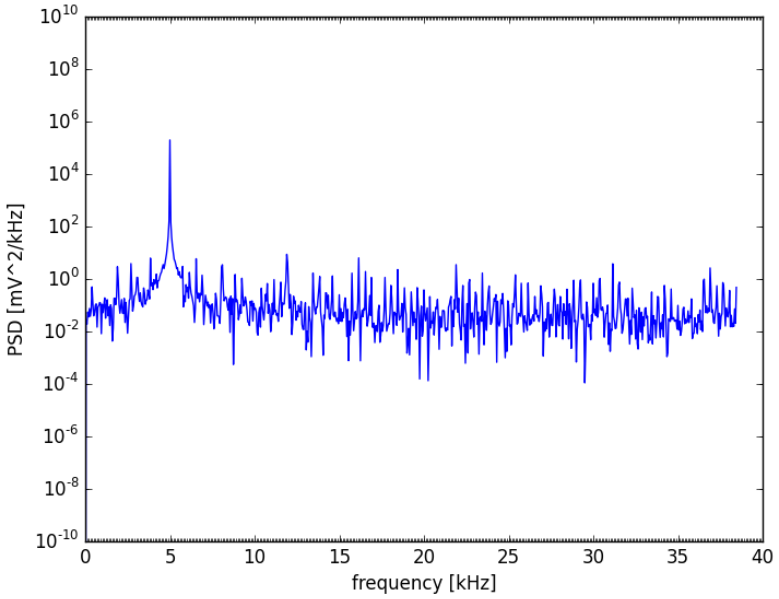


Figure 9: The python serial driver psd.py as seen at the command line (top) and the (bottom) corresponding power spectral distribution plot.

CircuitTester §

```
//
//  CircuitTester:  This is a stripped down sketch useful for testing
//     your circuit during construction... it continuously samples
//     the analog input and displays the mean and rms on the serial port.
//

// You should only have to edit these parameters, if at all...

const float CALIB_MV = 5000;  // Calibrated MV scale (nominal 5000 mV)
const int MAX_SCALE    = 220;  // Calibrated full scale: minimal clipping
const int MIN_SCALE    = 40;   // Calibrated min scale: ADC fails below
const int RUN_SCALE    = 80;
```

FastAdc §

```
//
// Fast ADC with PWM controlled ADC dynamic range adjustment.
//

// You should only have to edit these parameters, if at all...

const int MAX_SCALE    = 220;  // Calibrated full scale: minimal clipping
const int MIN_SCALE    = 40;   // Calibrated min scale: ADC fails below
const int RUN_SCALE    = 80;   // User selected value for PWM
const bool TEST_PATTERN = false; // Send test pattern instead of ADC data
//
// You shouldn't have to change anything below here...
//
```

Figure 10: The lab has two sketches. CircuitTester is essentially an RMS voltmeter that outputs directly to the Serial Monitor. You will calibrate CALIB_MV set initially to the nominal 5 V scale to match the actual full scale on your Arduino. The scale refers to the PWM output which determines the reference voltage for the ADC. A good all-around working point for this experiment is 80. The FastAdc sketch is used together with the psd.py python driver on your PC. The sketch only uses raw adc counts, but the calibration calib_mv in psd.py should match what you determine with CircuitTester. Apart from the parameters at the top of the sketches and driver, you shouldn't need to edit the code.

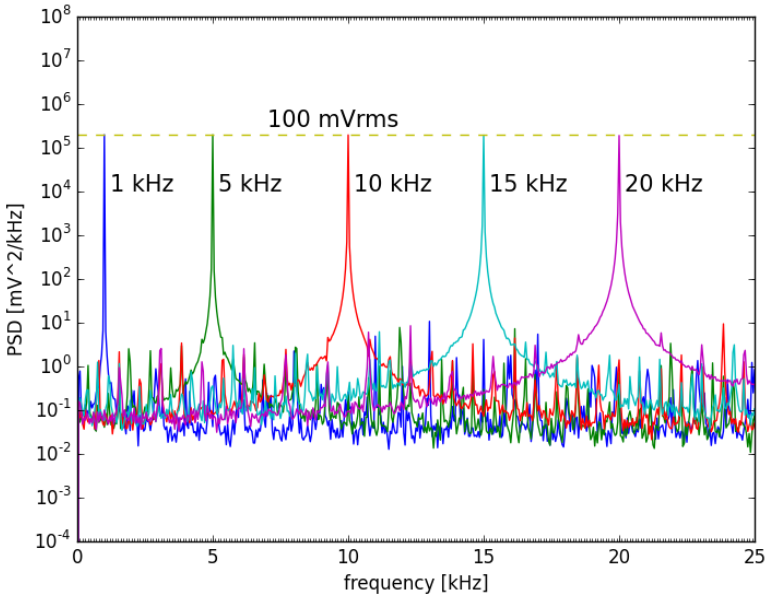


Figure 11: The response of the Arduino PSD.

9 Improved Johnson Noise Device Gain Measurement

The scope gain measurement is quite imprecise due to the presence of significant (few volts) low-frequency noise whenever the function generator is connected to the JN device. This noise is unambiguously due to an unavoidable (with your available test equipment) small ground loop between the function generator and your bench-top DC power supply. The tiny amount of noise due to this ground loop is amplified by the JN device by a factor of 1000. High gain instruments are a challenge! Fortunately, by moving this analysis into frequency space, we can largely avoid the low frequency noise in our measurement.

Using your Arduino spectrum analyzer, collect data for a range of frequencies, as shown in Fig. 12. You will use these samples to determine the gain of the JN device as function of frequency.

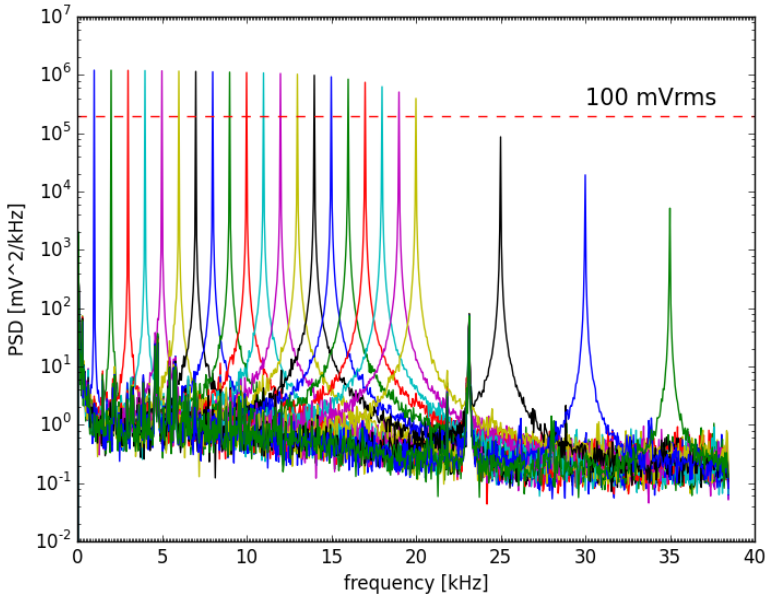


Figure 12: Data collected for gain versus frequency scan.

Unfortunately, the ground loop noise is not completely overcome by the power spectral analysis. The noise is so large that it saturates the ADC during some portion of the sample collection, effectively filtering out the higher frequency signal of interest, and systematically reducing our gain estimate.

Using an isolating transformer to break the ground loop by floating the function generator ground effectively eliminates this noise. Each JN device has been calibrated using this technique, and the measured gains at 5 kHz and 10 kHz are recorded below. The shape of your gain as a function of frequency can be inferred reliably from your data, but your should apply an overall scale factor to match one of the benchmarks at the benchmark frequency. This will give you a reliable Gain curve, accurate to about 5%.

Device	5 kHz	10 kHz
JN-A-#6	3.77	3.49
JN-A-#1	3.82	3.51
JN-A-#3	3.83	3.60
JN-A-#3	3.86	3.61
JN-A-#2	3.85	3.55
JN-B-#9	2.47	2.39
JN-B-#7	2.57	2.75

10 Johnson Noise Measurement

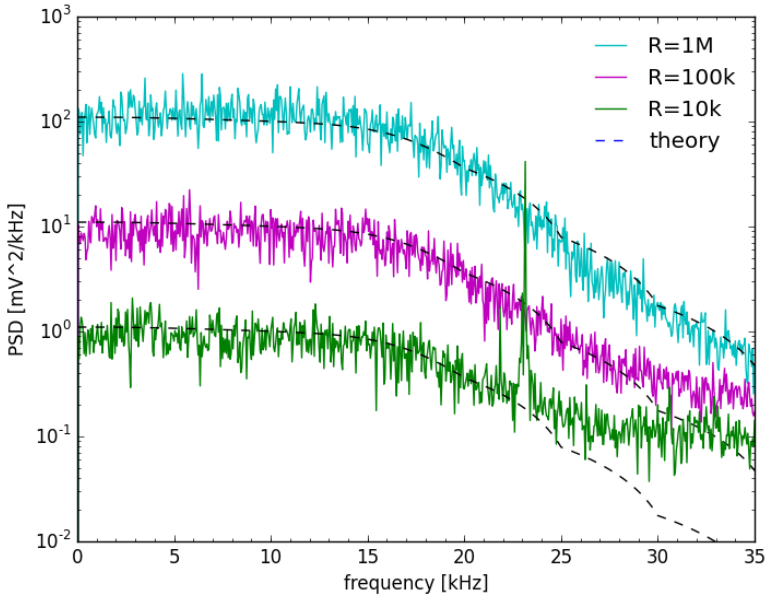


Figure 13: Data collected for Johnson Noise measurement.

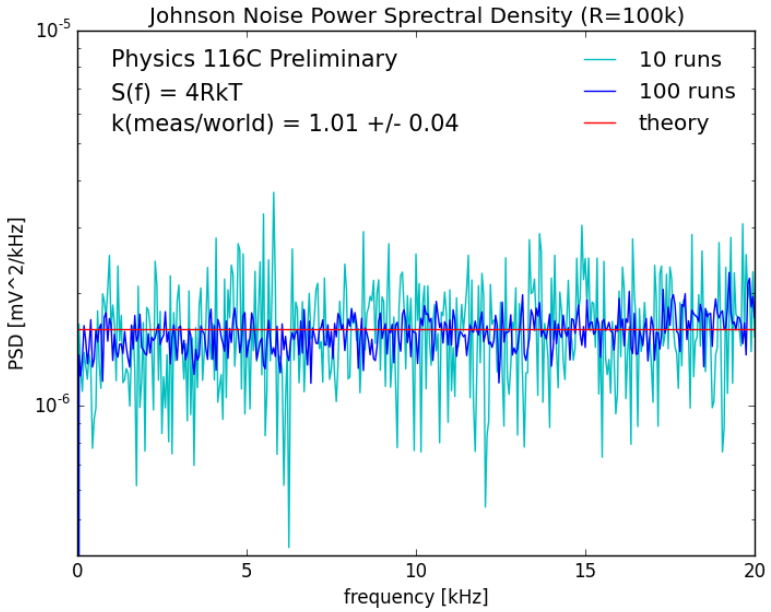


Figure 14: Instructor-level final plot.