This report will demonstrate how simple passive R-C filters can be utilized to reduce signal noise in automotive sensor signals. The knock sensor is used to detect “pinging” in modern cars in order to detect a lean air-fuel mixture ratio. This allows the ECU to retard the spark timing, increase fuel volume into the cylinders, or both. By reducing the spurious emissions, the computer control system can interpret the sensor signals more accurately and reliably.

Automotive Knock Sensor Filtering

How to use reduce sensor signal noise with RC Low Pass filters.

Ronald P. Kessler, Ph.D., Advisor for UCI Race Car Teams

Table of Contents

Introduction

Reading analog sensor data in automotive environments is a real challenge. Engine noise, alternator whine, fuel injector pulses, and ignition spark all contribute to the collection of unwanted electro-magnetic interference (EMI) that plagues our sensor readings. These unwanted frequencies degrade and mask the sensor data we are attempting to collect, interpret, and display to the operator. The data must be robust. Collision-avoidance sensors cannot simply work most of the time, for example. They must work all the time.

This paper will explore the possibility of utilizing hardware filtering circuits to achieve the goal of removing EMI and preserving the integrity of our readings. This approach is presented as an alternative to software filtering strategies. It is strongly believed that removing electrical noise at the source is superior to the use of software algorithms alone. However, in some cases, it may make sense to use both. In any case, we will focus on how we might process raw sensor readings so as to make microcontroller interpretation of the readings more consistent.

The sensor we will look at here is the knock sensor (Figure 1). This sensor is used to notify the Engine Control Module (ECM) that the engine is "pinging". When an engine runs too lean, the fuel is burned pre-maturely (detonation) and the driver hears a ping or rattling/knocking noise while going up a hill, for instance. Left unchecked, the engine can overheat and eventually burn a hole in the piston.

Figure 1: Typical Knock Sensor

When a lean condition is triggered by the sensor, the ECU will add fuel to the engine or retard the ignition timing (or both) in order to mitigate this situation. Automotive engineers typically use a "knock sensor" made from a piezoelectric crystal device. These devices are used in industrial manufacturing, medical imaging, and are even used to detect flaws in aircraft engine components. They work in a rather simple manner. Let's see how.

These sensors detect irregular engine vibrations caused by this pinging. As such, they are similar to accelerometers. Figure 2 shows the signal from my testing. Notice that there is one large pulse as the sensor is tapped against the workbench. You can see a lot of other signals as the vibration decays.

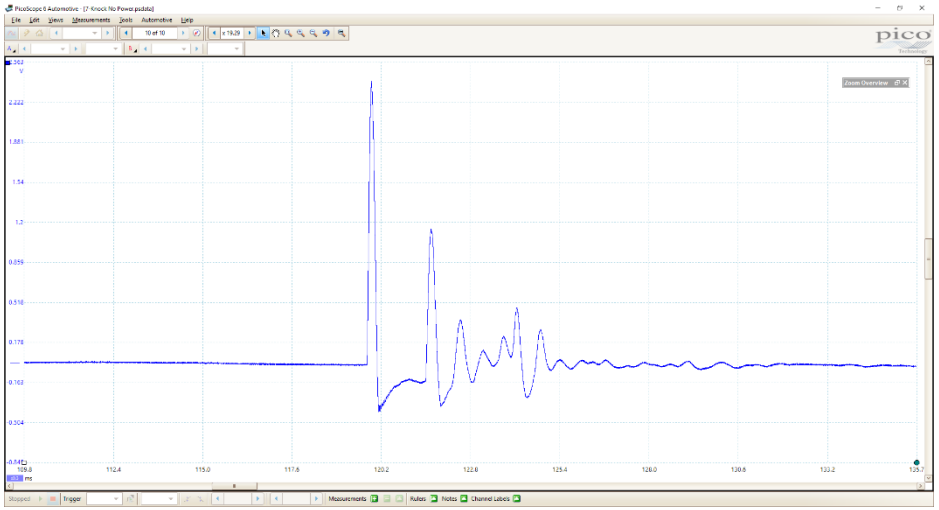

Figure 2: Knock Sensor Signal After Being Tapped on The Workbench

Special ceramic discs are used that respond to pressure or vibration. When detonation occurs, the vibrations are telescoped through the engine block and picked up by this sensor. The vibrations place pressure on the disc. The disc creates an AC voltage which it sends on to the ECU for interpretation. The output signal is proportional to the pressure or vibration the sensor detects.

They are called signal generators because they create their own signal without the need for a reference voltage. Manufacturers tune these sensors to be sensitive to the frequency that corresponds to a detonation/pinging condition in each of their engines. Thus, they are application specific. It turns out they are very accurate and reliable devices.

Bench Testing a Knock Sensor

To test one of these sensors, I connected it to my Picoscope as shown in Figure 3. There are two connector pins. I connected my Picoscope probe to one pin and the probe's ground to the other. I then lightly tapped the sensor repeatedly on the bench to simulate pinging.

Figure 3: Bench Testing the Knock Sensor

Figure 4 shows the results of that test in the scope image. You can see the large spikes and the "ringing" between those spikes as it was being tapped. Remember, it is not necessary to have any power connections for this test. These devices create their own current flow without the need for any external power.

Each spike measures the gentle force of the sensor striking the bench. From these images, it is easy to see why a microcontroller might have difficulty interpreting the results. Wouldn't it be nice if we could "clean" up the signal so the extraneous noise from those small ringing frequencies would be eliminated or at least minimized?

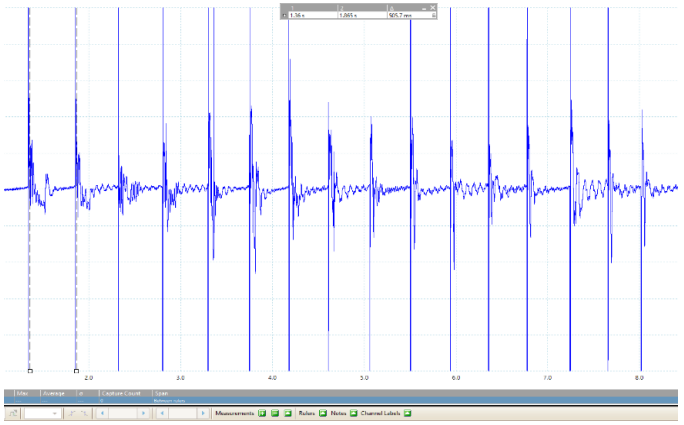

Figure 4: Knock Sensor Signal Created by Tapping on It Against the Workbench About 2 Times/Second

Experiment 1: Process Signal Data with Single-Pole Hardware Filters

To demonstrate what I would like to do, I want to show you another zoomed-in image that shows those unwanted frequencies. Figure 5 shows the expanded scope image from my original image (Figure 4). I measured some of the noise pulses using the vertical cursors (dotted vertical lines) and discovered they ranged in frequency from, 35-2000Hz.

Let's see if we can build a filter to clean up this signal. Since the frequency of my signal was around 2Hz (caused by the speed of my tapping), I decided to filter out all frequencies above 15Hz to see what would happen.

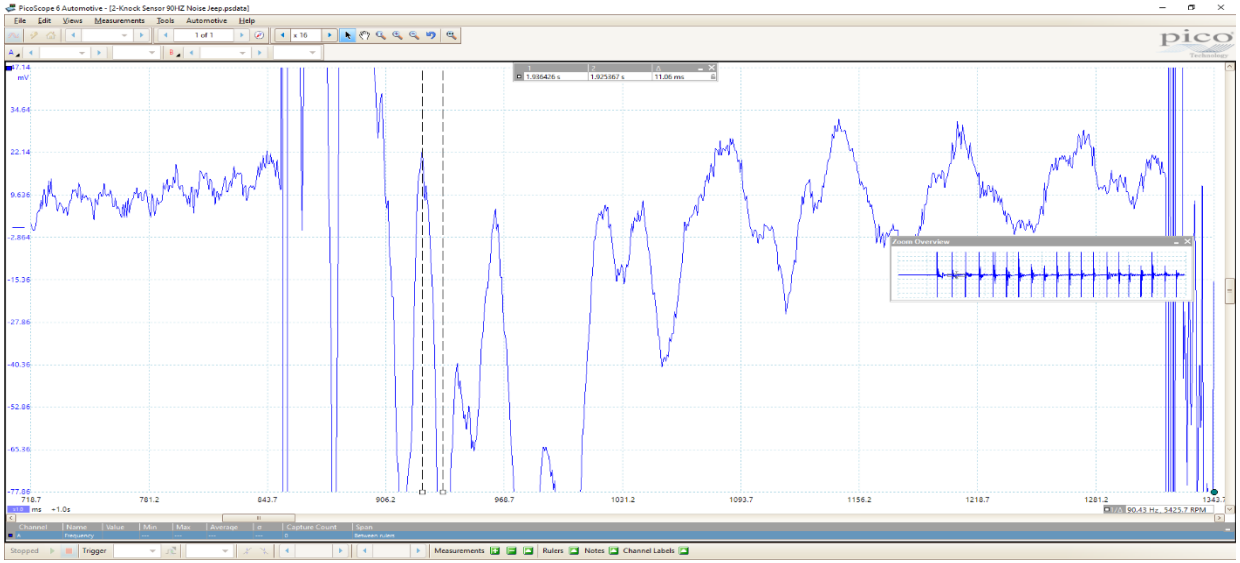

Figure 5: Expanded View of the Signal in Figure 4 That Shows 35-2000Hz Noise Embedded in the Desired Signal.

However, if you were designing a filter for a real application, you would have to determine the maximum frequency your sensor would produce and filter out all frequencies above that. For example, a wheel or gear rotation sensor needs to pick up the maximum RPM the object will rotate. Once that is determined, you can filter out all frequencies above that.

Now it is time to design our filter.

Designing a Low Pass Filter in National Instruments Multisim

In order to make this signal easier for the MCU or controller to decipher, I decided to apply a single-pole, passive, low-pass RC filter using one resistor and one ceramic (non-polarized) capacitor. The single-pole refers to the fact we are using only one reactive component, the capacitor, that is sensitive to frequency changes. The resistor is not. Filters that use two capacitors are called two-pole because they use two reactive components, and so on. You can use several stages or poles in your design. These filters are described in detail in my article entitled *Exploring Sensor Signal Processing*.

In my test scenario, the pulses I want to preserve occur at around 2Hz or so in this case. Recall, that just happened to be the speed at which I tapped the sensor. So, I decided that any frequency above 15Hz was just noise or unwanted pulses. Keep in mind, filters are not perfect. You cannot build one that passes 2Hz signals but absolutely blocks 3Hz signals and above. That is why I chose a frequency slightly larger than the signal I want to preserve. Even though our components are not super accurate, you *can* improve filters by using multi-stage designs. We will check that out in a minute.

My filter setup from the Multisim simulator is shown in Figure 7. The schematic shows how the input signal is connected to the resistor and we measure the output across the capacitor. This is what makes up a low pass filter. At low frequencies, the capacitor will allow low frequencies to pass but block high frequencies (hence low-pass filter). The point where the pass-no pass frequency happens is based on the values chosen for R_1 and C_1 and is called the “cutoff” or “corner” frequency of my filter. The cool thing is, we can choose the cutoff frequency for any particular project we are working on! By the way, if we reversed the position of R_1 and C_1, we would have a high-pass filter. In that case, low frequencies would be blocked and only high frequencies would pass through.

I used a 10V Peak-to-Peak sine wave at a frequency of 2Hz to represent my sensor signal I observed while tapping it. To create a filter with the 15Hz cutoff, I chose a 1000Ω (1K) resistor and 10µF (microfarad) ceramic capacitor. Calculators are online for us to determine this cutoff frequency (f_c).

The formula is: $f_c = \frac{1}{2\pi RC}$

The components I chose should pass frequencies from 1-15Hz but will start blocking or attenuating signals above that.

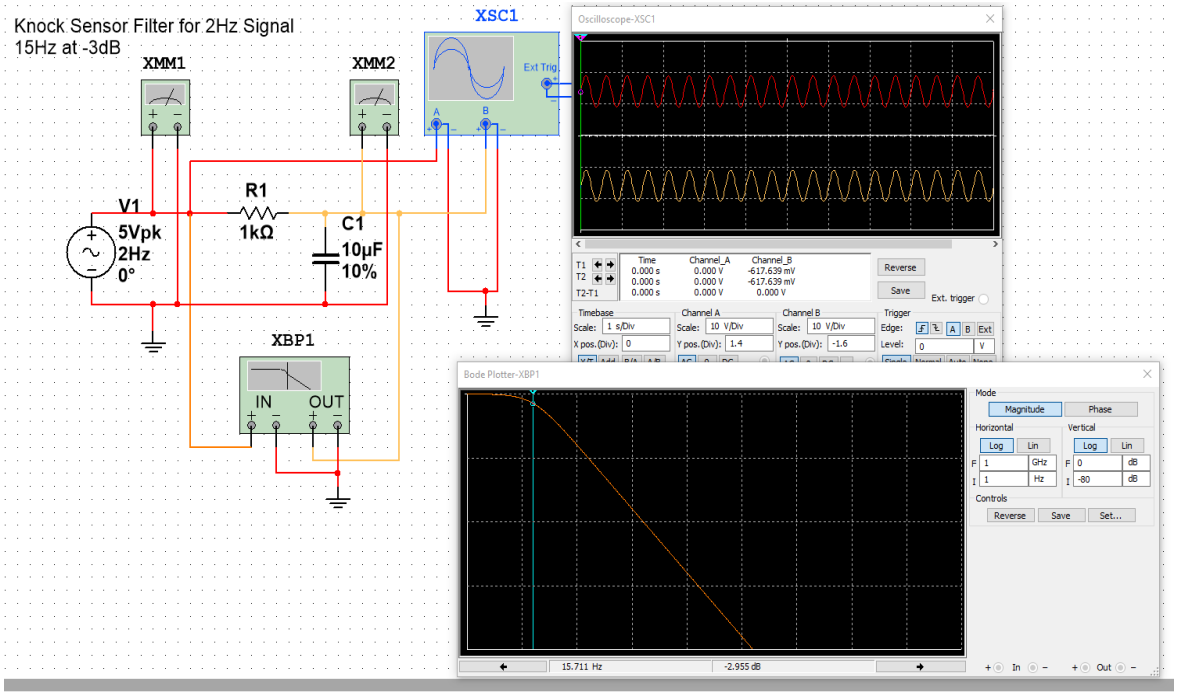

Figure 7: RC Low-Pass Filter Simulation. The Red Trace Shows Input Signal and the Yellow Shows the Output. At 2Hz, the Output Signal is Passed Through the Filter Easily. The Bode Plot Shows the Frequency Response and the Output Voltage of The Filter. The Chart Illustrates How Frequencies Above 15Hz Are Dramatically Reduced/Attenuated.

If you look at the scope image, the red trace shows my 2Hz input signal. The yellow trace shows the output of the filter. Notice the amplitude of the two traces are nearly equal. This demonstrates how easily my low frequency signal (2Hz) passes through. In other words, the amplitude (voltage) of the output is almost identical to the amplitude of the input signal. The frequency response curve or Bode plot in the second scope image shows how signals above my cutoff frequency are blocked. The graph shows the output voltage dropping quickly after cutoff (the green vertical line). This means those high frequency signals I identified in the 30-2000Hz range will be reduced in strength and mostly eliminated. This should make our sensor signal much easier to process with an analog to digital converter (ADC), for example.

Applying the Low Pass Filter

Now it is time to compare the original signal (Figure 4), with the filtered or unwanted noise image shown in Figure 8. The signals are significantly cleaner. With the filter in place, the sensor pulses should be more easily detected by my microcontroller! Remember, this filter allows frequencies *below* 15Hz to pass through freely while frequencies above that are attenuated.

Figure 8: Knock Sensor Trace With 15Hz Low Pass Filter Installed

Experiment 2: Using a Two-Pole Filter

Before we finish up our discussion of filters, I wanted to mention that designers often use two filter stages so as to improve filter functioning. The design shown in Figure 9 is a two-stage or two-pole low-pass filter. It is made up of *two* single stage filters. In this circuit, I now use two resistors and two capacitors. Since the capacitors react to changes in frequency (resistors do not), there are now two reactive components (the capacitors) and that is why it is called a two-pole filter. These designs are used when the roll-off characteristics of a single-pole filter are not quite good enough to remove unwanted noise. The second stage picks up where the first stage left off and attenuates our signal as a higher *rate*.

Thus, the second pole/stage causes the steeper roll off in the graph and removes more unwanted frequencies just after cutoff. This steeper roll-off attenuates signals at 40dB/decade compared to 20dB/Decade.

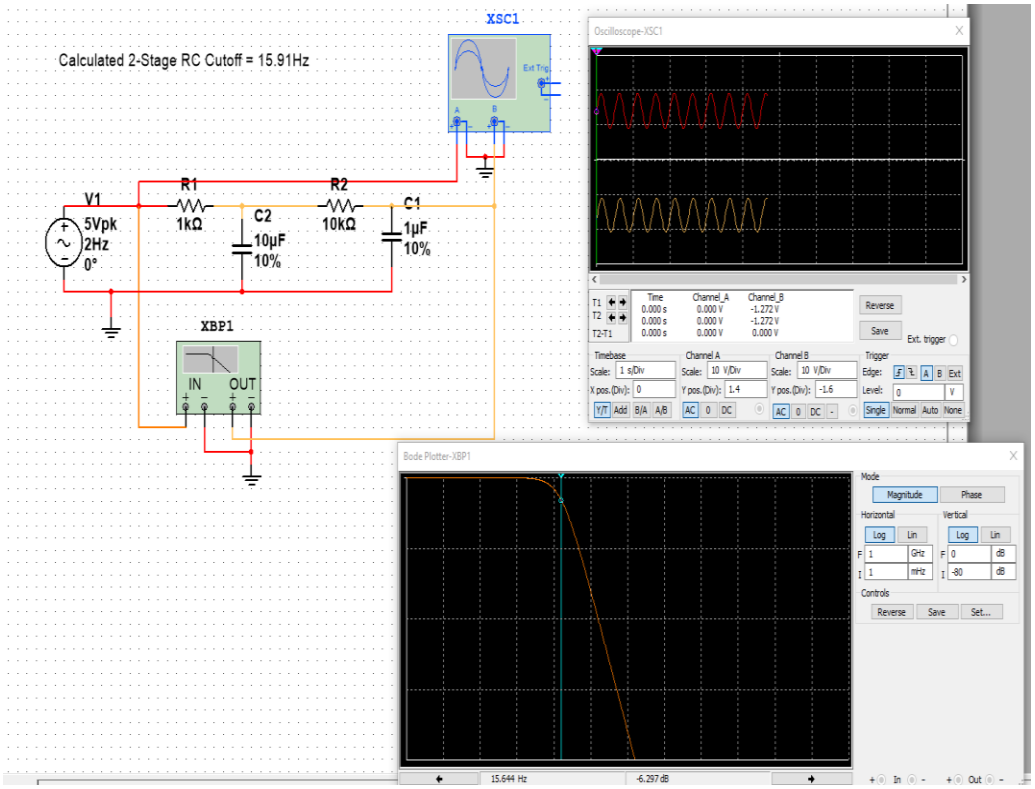

Figure 9: Two-Pole (two-stage) RC filter design. Notice the roll off of the filter is twice as steep as my original design.

With more components, our formulas change slightly. For example, to compute the cutoff frequency for 2-pole filter:

$$f_c = \frac{1}{2\pi\sqrt{R_1 C_1 R_2 C_2}} \text{ Hz}$$

When two poles are used, the decrease in signal strength at cutoff is twice that of the single-pole design which is -3dB. So, at the calculated cutoff of 15.91Hz, the two-pole design will reduce the signal by -6dB. Hence, the graph is twice as steep. The term *decade* refers to the fact that the x-axis on my graphs are in logarithmic scale. The frequency is plotted as 0, 10, 100, 1000, 10,000, 100,000 Hz.

Simulation vs Actual Filter Responses

Single-Pole Design

For clarity, let's compare the frequency response for both filters side by side. You will discover that the simulator results and your actual readings will be different because the quality of our physical components are not perfect.

We calculated the cutoff formula for the 1-Pole filter as 15.9Hz. However, Figure 10 shows the actual reading is slightly lower.

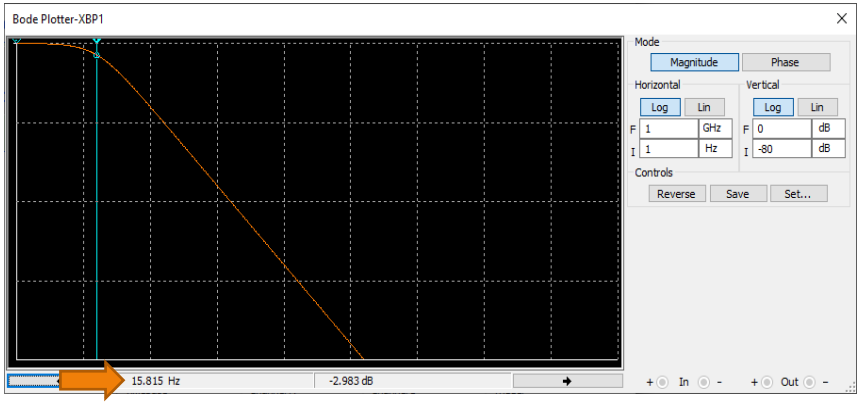

Figure 10: Bode Plot for Single-Pole Design

Two-Pole Design

With the use of a 2-pole filter, the descrepancy is more pronounced. This is due to the effects of the added resistor and capacitor. It actually lowers the cutoff frequency.

We calculate the 2-pole cutoff value with a slightly modified formula. At -3dB, the cutoff (f_{-3dB}) is calculated as:

$$f_{-3dB} = f_c\sqrt{2^{\left(\frac{1}{n}\right)} - 1}$$

$$= \mathbf{15.91}(\sqrt{1.41 - 1}\,)$$

$$= 15.91(.64)$$

$$= \underline{\mathbf{10.18Hz}}$$

where *n* = the number of filter stages.

Figure 11 shows the *actual* cutoff frequency is now 9.7Hz.

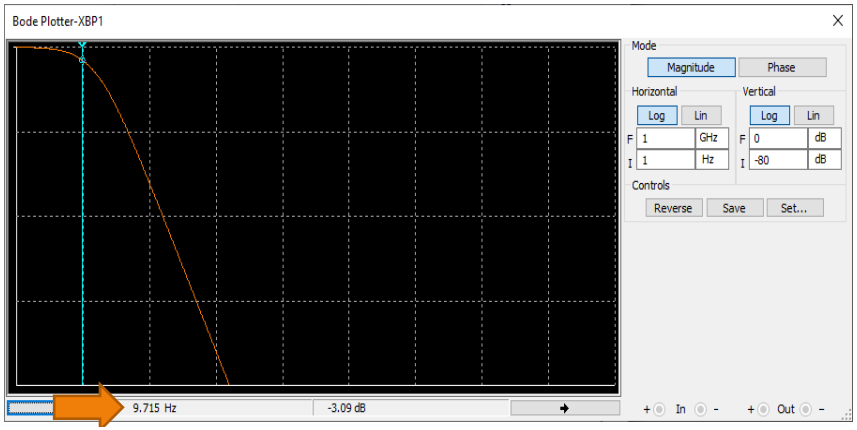

Figure 11: Bode Plot for Two-Pole Filter. The Roll Off is Twice as Steep as the Single-Pole Design (40dB vs 20dB/Decade)

Finally, it is important to remember that the frequency response of these filters will vary with the resistance of the load or the circuit you are connecting them to. Most textbooks use a 50Ω load when they show simulations. So be sure and test your circuit with that in mind.

Summary and Conclusions

Analog signal processing is a difficult but necessary element of telemetry systems in many industries. We have explored the use of hardware filtering techniques that can help improve the accuracy and reliability in automotive sensor signals. While this paper only focused on knock sensors, the results can be generalized to many sensor circuits including Hall Effect Crank/Camshaft sensors, IR, accelerometer, and RPM monitoring systems.

The use of passive RC filters appears to be a viable solution to the problem of how to eliminate EMI in low voltage sensor circuits. It is a good idea to remove unwanted noise at the source instead of using software algorithms such as moving average and exponential moving average. While the software solutions are quite effective in many scenarios, hardware filters should be strongly considered when gathering sensor readings in noisy industrial and automotive environments. We have seen how both single-pole and two-pole (two-stage) filters can be quite effective at rapidly attenuating unwanted frequencies in the “stop band” of RC filters. Keep in mind however, the more stages/poles that are added to the filter, the more the output signal is reduced. When the output needs to be amplified so as to accommodate analog-to-digital converters (ADC), active filters using op-amps should be used so the gain can be calibrated to match the needs of the microcontroller.

The next step in this journey would be to actually process our filtered data with a microcontroller and compare the reliability of the readings to non-filtered and software-only filtered data. Stay tuned!