The Curious Case of Short Reference Signals: A Debugging Session

Tanner Bitz

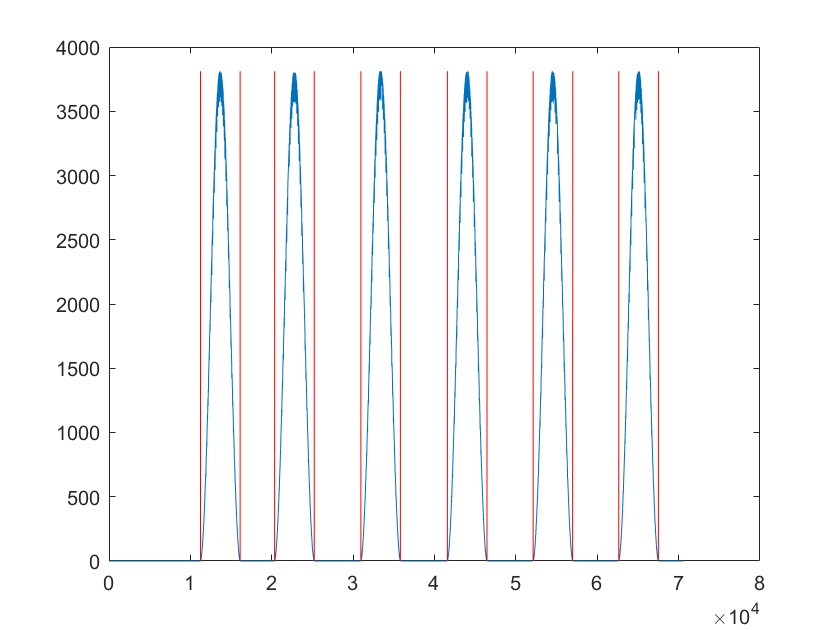
1/28/19

*Background:*

Dr. Lee noticed that the 0.2 Hz reference signal was shorter than expected. It should last 5s, which should correspond to 5000 samples on the DAQ. Upon first inspection it appeared to be 4700-4800 samples.

*Looking A Little Further:*

I analyzed all 0.20Hz DF data files from Pat50, Pat51, and Pat52 to look at the length of the reference signal. To determine the length of the cycle I determined the first and last indices where the reference signal data was non-zero. The plot below gives an idea how well this determines the reference signal period.



There 6 reference signals a file and 13 files total, for a total of 78 reference signals. Each cycle is supposed to last 5s, which should be 5000 samples. Looking at all 78 reference signals, there are a mean of 4841 samples, with a standard deviation of 32.51 samples.

This corresponds to a 968 +/- 6.50 samples/sec.

The MATLAB analysis script is named: analyze0\_2HzReferenceSignalLength.m

*Problem*

Why is the reference signal measuring a shorter period than expected?

*Possible Causes*

1. DAQ isn’t sampling at 1000Hz.
2. MCP4725 (DAC) from NodeMcu isn’t generating enough voltage at the beginning and end of the signals for the resolution of the DAQ to pick it up. Thus a portion of the beginning and end of the cycle would be seen as “cut off” by the DAQ.
3. NodeMcu (wifi module) has a clock speed that doesn’t allow for it sample at 1000Hz. It could think it’s going through 5000 cycles of signal generation, but it maybe running fast, thus making the 5000 cycles duration last less than 5 seconds.

*Investigation*

*Cause 1: DAQ Isn’t Sampling at 1000Hz*

I wrote a python script that inserted an integer in the DAQ readings every sec as well as printed out the time. In each of the three separate trials, I sent a new number as a flag one second apart 60 times (so there are 59 one second intervals per trial, 177 intervals in total). These intervals were, on average, 999.999ms apart, with a std of 0.112ms. Effectively one sec apart every time. During these one second intervals the DAQ took an average of 999.95 samples with a std of 0.582 samples. Effectively, this is running at 1000Hz. A histogram of the number of samples in these one sec intervals can be seen below.

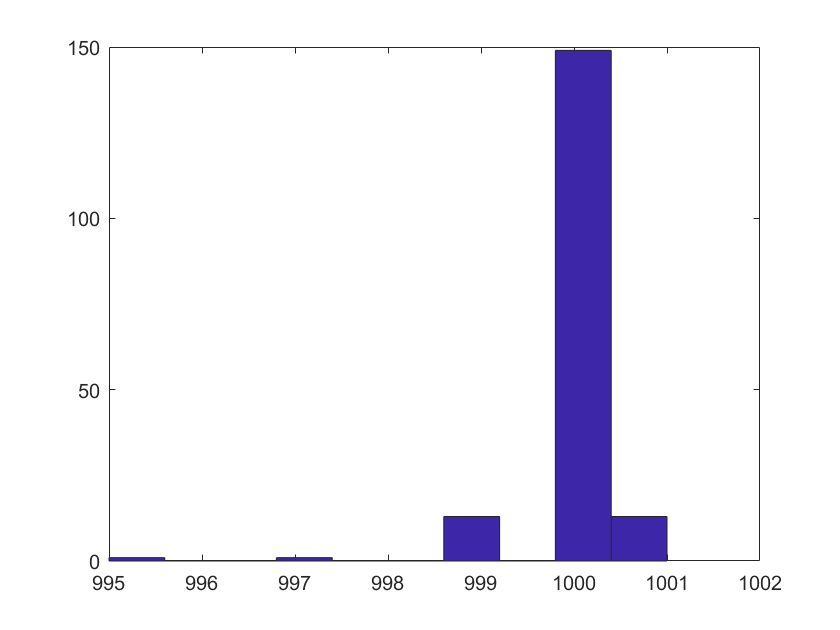


Figure 1: Histogram --Samples In A One Sec Interval

Python test script: testDAQ.py

MATLAB analysis script: analyzeDaqSampleSpeed.m

*Cause 2: DAC Isn’t Generating Enough Voltage At Beginning/End of Cycles To Activate Smallest DAQ Resolution*

Hardware:

DAQ – DAQ127 (MAX127 chip)

DAC (I2C Slave Device) – MCP4725

Microcontroller (I2C Master Device) - NodeMcu

Setup Description

1. NodeMcu receives a http GET request from the GUI or another python program, which triggers the NodeMcu to switch to change the sinusoid freq to whatever was received.
2. NodeMcu receives a http GET request from the GUI or another python program, which triggers a timer to start at 1000Hz and be called until one full cycle has written. For instance, if the freq is changed to 0.2Hz (5s period), the timer is triggered 5000 times before it turns off.
3. During each of these timer triggers, a value corresponding to a voltage on our 12 bit DAC is calculated using the below formula:

For instance, lets say the freq is 0.2 Hz and we want to know the value generated at 1sec (1000ms – theoretically at timer\_count = 1000):

1. This value is sent to the MCP4725 which is a 12-bit DAC that outputs 0-5V theoretically. The output of this DAC is measured by the DAQ at 1000Hz using a 12-bit ADC for 0-5V theoretically.

During setup of the DAC a maximum 12-bit value in streamed from the NodeMcu to the MCP4725, which should generate a 5V signal and should be read into the DAQ as 4095 (the maximum 12 bit read from the DAQ). Typically, however the value is read in at about 3800. Given this, one would expect the entire MCP4725 to be linearly shifted down. This leads to the question, when is the first time in a sinusoid period one would expect to see a nonzero read on the DAQ? Let’s assume that 5V is being generated perfectly from the MCP4725 (its probably not, but it’s the proportions in these equations that matter)

DAQ Resolution --- 5V/3800 = 0.001315789473684 V/ discrete val

NodeMcu --- 5V/4096 = 0.001220703125000 V/discrete val

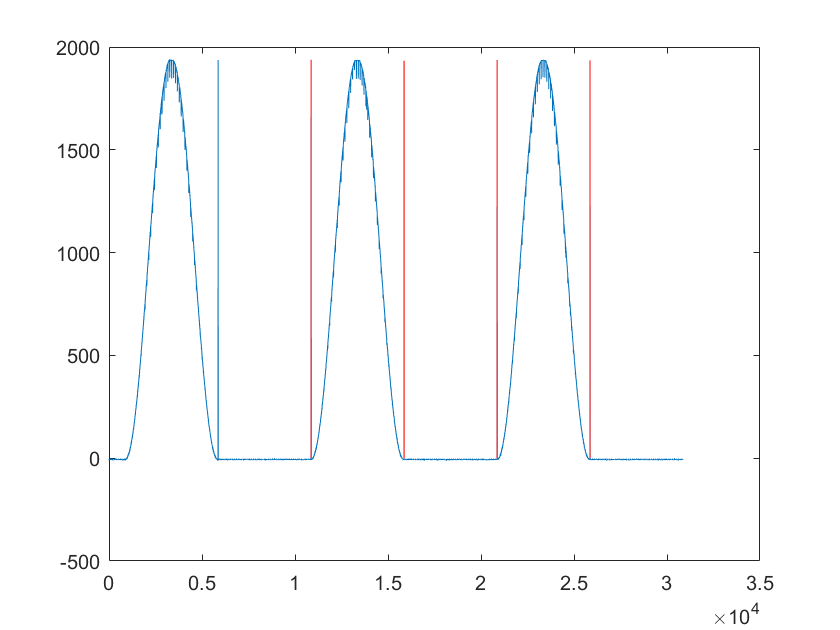
This would mean the DAC would have to get a value of 2 from the NodeMcu to generate a signal large enough to register with the DAQ’s resolution. When would that first happen in a sinusoid cycle?

In a 0.2Hz sinusoid, the first time this would occur is at 6ms. So it would have sense to see 6ms cut off both ends of the sinusoid (12 total cut off).

This doesn’t account for ~159 samples that we are seeing cut off (on average).

*Cause 3: NodeMcu Timer Isn’t Acting At 1000Hz.*

To test this, I made a method in the NodeMcu code that ran a normal unidirectional cycle (like it normally does) and put 5ms of full voltage before and after the cycle. This way I would see a spike in the data on either side of the signal. I would then be able to look at the number of samples the DAQ took and determine if it was running at the correct frequency. Below is a picture of a 0.2Hz signal with the spikes on either side.



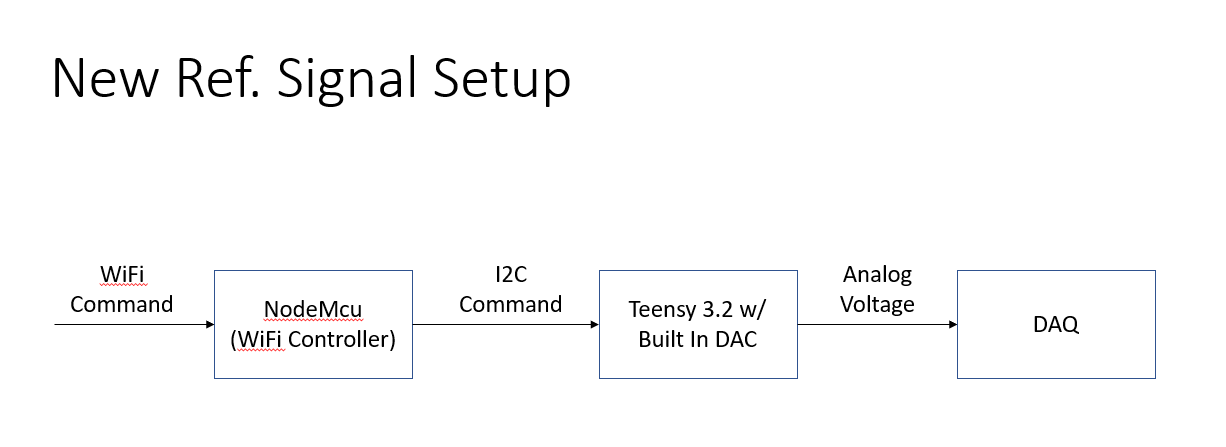
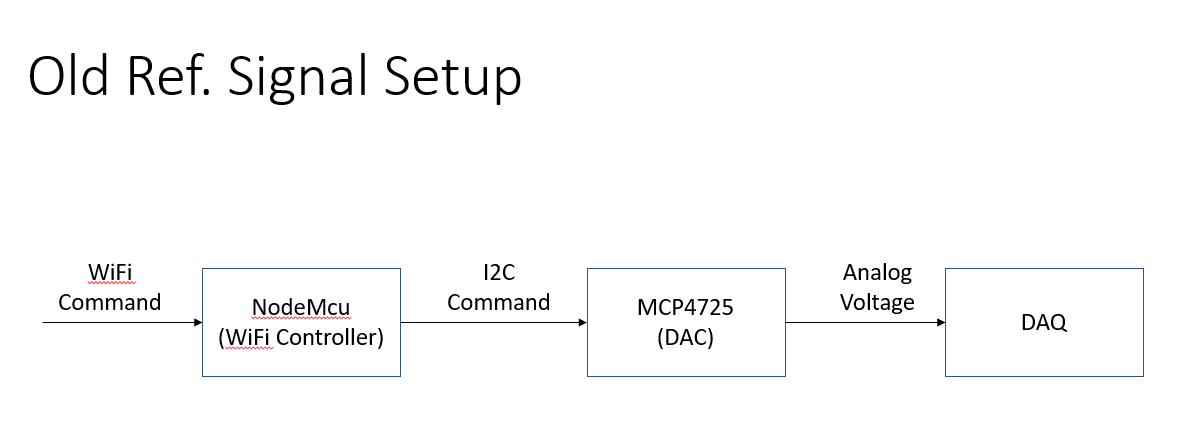
At this point, I was able to see for these two cycles that they were close to 5000 samples long. The first starts at sample 10870 and ends at 15859, which is 4990 samples long, and the second cycle starts at 20869 and ends 25860, which makes it 4992 samples long.

At this point, a happy accident happened. Usually when the DAC is being read into the DAQ the DAQ is set to 0-5V. For this trial it was set for -5 to +5V. This is noteworthy as there were measurements that went as negative as -12, which means the DAQ had a 12\*10V/4096 = 0.0293V negative bias. When this is taken into account it could explain why samples were getting cut off the start and end of the signal.

*Action Taken*

There were two outcomes of this investigation.

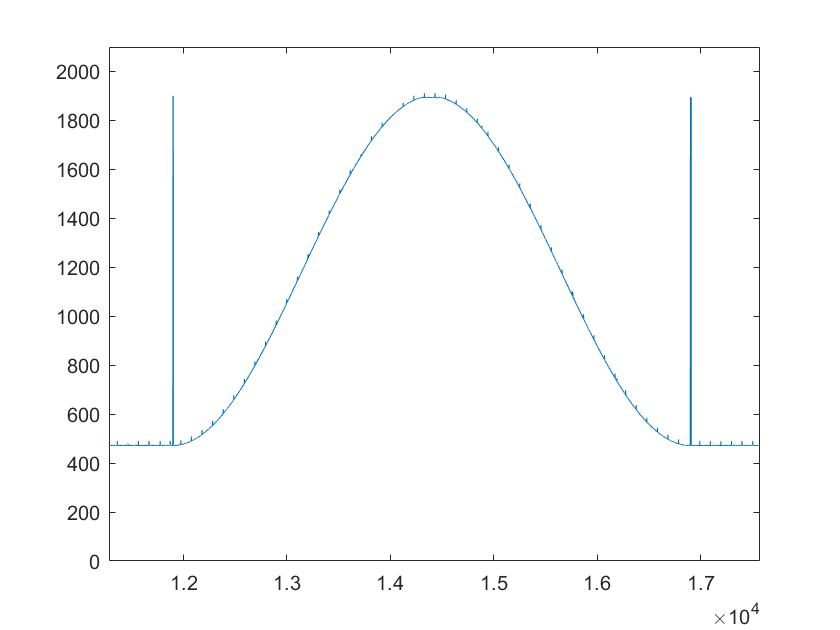
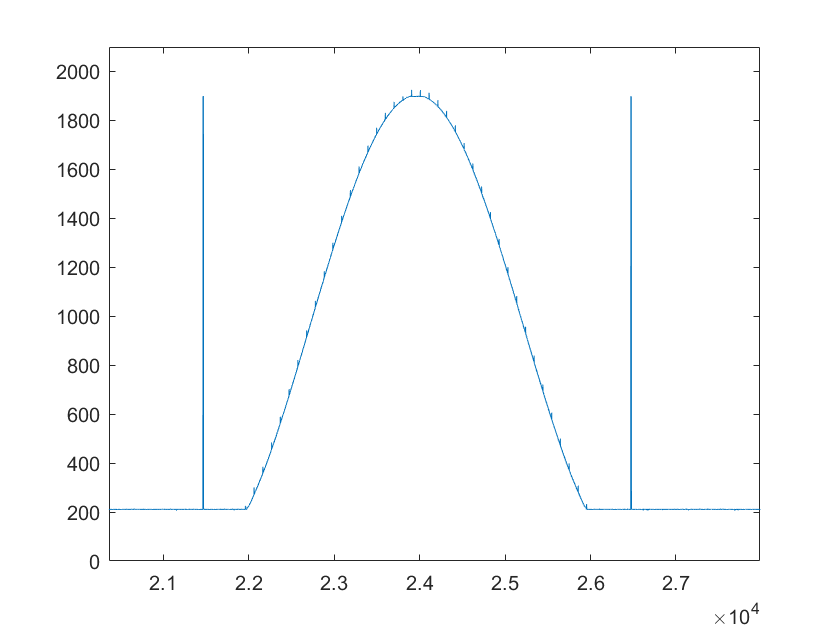
1. The NodeMcu’s timer was determined to have insufficient resolution on interrupts. Based on this the setup of the reference signal was changed.



1. The floor and ceiling voltage of the sine signal can be changed with a Wifi Command. This allows us to choose a range of output from out DAC that doesn’t saturate when it is read in by the DAQ.

*Results*

The saturation and variability in number of samples/cycle has been removed.



Before: Saturation After: No Saturation