Name:	Major: ☐ ELEC ☐ CSEN

ECEG 201 Laboratory 5, Calculating Better Averages

Introduction

In this laboratory activity you will dig deeper into the notion of averaging noisy data. You will try to determine how many samples you should average from the Feather's ADC in order to reduce the noise to an acceptable level. You should read through this entire handout, and make sure you understand what you are trying to accomplish, before starting to gather data.

Deliverables

You must turn in this worksheet by the beginning of class on Wednesday, 2020-03-04. Each student will work individually on this activity. You may discuss the activity with other students but you must do your own work and everything you submit for grading must be entirely your own work.

You must send the instructor a copy of the Python code that you used on the Feather for this lab. Remember that the Feather shows up on the desktop like a flash drive, and you can just copy your `code.py` or `main.py` directly from it. Attach the Python source file to an email message; do not import it to a word processor file or modify it in any way. The file you send must be suitable for execution on a Feather with no changes. You should follow the formatting and style guidelines give in PEP 8 *Style Guide for Python Code*. At the very least, you should use the `Check` button in Mu and fix anything that it complains about. The deadline for submitting your code is the same as that for this worksheet.

Procedure

We have made the assumption that the noise appearing in the Feather's Analog-to-Digital converter output is random, and that we can reduce the amount of noise in our readings by **averaging** some number of ADC digital values. The question we want to answer today is: how many samples should I average in order to reduce the noise to an acceptable level? We are going to try to answer this question by computing the average several times, and then looking at the statistics of the **averages themselves**.

First, you need to write code for the Feather that reads the raw integer value from the ADC and then averages some fixed number of these readings. The value of the "fixed number" should be assigned to a constant named `SAMPLES_TO_AVERAGE` at the top of your Python module, then you would use this constant in some kind of loop structure to make readings and compute the average.

After you read each integer sample from the ADC, print that value out to the console in the Mu editor. This will also cause a small delay between samples, since the ADC can't produce new samples as fast as your code could read them.

Second, you will compute some number of these sample averages and then compute the average, minimum, and maximum value of those averages. The value of "some number" should be assigned to a constant named `AVERAGES_TO_AVERAGE` at the top of your module. Once your program has calculated that many averages it should print out the average of the averages, the minimum of the averages, and the maximum of the averages. Print all of the these values on a single line, and add some text to indicate which value is which. All of these calculations should be done on the Feather.

By looking at the values for the average of averages, the minimum of averages, and the maximum of averages you can get a sense of how **precise** or **repeatable** your average-of-ADC-readings is. If there is a large difference between the minimum and maximum values then it indicates that your average calculations are not very good, and you probably need to increase the value of `SAMPLES_TO_AVG`.

So how good is good enough? How small should the difference be between the minimum average and the average average, and between the maximum average and the average average? Remember that for any ADC we must accept that the **quantization error** is $\pm\frac{1}{2}$ LSB. We also know that the Feather's ADC actually produces a 12-bit result, but the `value` method returns this 12-bit result left-justified in a 16-bit unsigned integer value…the right-most four

bits are always zero. So, if the LSB of the actual 12-bit ADC result was to change from a 0 to a 1 (or vice versa), how much would the number returned by `value` change? If you average enough ADC readings can you achieve a precision such that the averages you calculate do not vary more than $\pm\frac{1}{2}$ of the change that would be caused by changing the value of the ADC's LSB?

Use the Analog Discovery to provide an input voltage of (1.00 ± 0.01) V to one of the Feather's analog inputs. Measure the voltage using the Analog Discovery and **record** its value here. Calculate and **record** the value you would expect to get from the `value` method if the Feather's ADC was ideal.

As a starting point for your experiment, try setting

```
SAMPLES_TO_AVERAGE = 4
AVERAGES_TO_AVERAGE = 16
```

Calculate 16 averages, where each average is the average of 4 ADC readings. For these 16 averages, calculate the average of the averages, the minimum of the averages, and the maximum of the averages. If the difference between the average of the averages and the minimum of the averages, or between the average of the averages and the maximum of the averages, is greater than the quantization error then **double** the value of `SAMPLES_TO_AVERAGE` (but do not change `AVERAGES_TO_AVERAGE`). Continue until the precision of the averages is less than the quantization error. Record your observations in the table below. You may not need all of the rows.

SAMPLES_TO_AVERAGE	Average of Averages	Minimum of Averages	Maximum of Averages
4			