Example Program: sw polled data logged to csv file

Here's an example app that logs software polled data to a csv file.
The data is collected using a timer (default is 0.5 second interval but can be set to another value) and demonstrates writing data in Python.
the file name is auto generated based on date and time for example:
yyyy_Month_dd_hhmmss.csv
The data is stored in $ /home/pi/Documents/Measurement_Computing/log_files

a time stamp is generated at every iteration of the timer tick including date and time to the second.

```
 #!/usr/bin/env python
#  -*- coding: utf-8 -*-
"""
    MCC 118 Functions Demonstrated:
        mcc118.a_in_read

    Purpose:
        Read a single data value for each channel in a loop.

    Description:
        This example demonstrates acquiring data using a software timed loop
        to read a single value from each selected channel on each iteration
        of the loop.
"""
from __future__ import print_function
from time import sleep
from sys import stdout
from daqhats import mcc118, OptionFlags, HatIDs, HatError
from daqhats_utils import select_hat_device, enum_mask_to_string
from datetime import datetime, timedelta

import os
import csv    #this is the import to make csv file creation simple.
import errno

# Constants
CURSOR_BACK_2 = '\x1b[2D'
ERASE_TO_END_OF_LINE = '\x1b[0K'


def main():
    """
    This function is executed automatically when the module is run directly.
```

```
    """
    options = OptionFlags.DEFAULT
    low_chan = 0
    high_chan = 3
    mcc_118_num_channels = mcc118.info().NUM_AI_CHANNELS
    sample_interval = 0.5  # In Seconds
    basepath = '/home/pi/Documents/Measurement_Computing'
    mypath = basepath + '/log_files'

    try:
        # Ensure low_chan and high_chan are valid.
        if low_chan < 0 or low_chan >= mcc_118_num_channels:
            error_message = ('Error: Invalid low_chan selection - must be '
                             '0 - {0:d}'.format(mcc_118_num_channels - 1))
            raise Exception(error_message)
        if high_chan < 0 or high_chan >= mcc_118_num_channels:
            error_message = ('Error: Invalid high_chan selection - must be '
                             '0 - {0:d}'.format(mcc_118_num_channels - 1))
            raise Exception(error_message)
        if low_chan > high_chan:
            error_message = ('Error: Invalid channels - high_chan must be '
                             'greater than or equal to low_chan')
            raise Exception(error_message)

        # Get an instance of the selected hat device object.
        address = select_hat_device(HatIDs.MCC_118)
        hat = mcc118(address)

        print('\nMCC 118 single data value read example')
        print('    Function demonstrated: mcc118.a_in_read')
        print('    Channels: {0:d} - {1:d}'.format(low_chan, high_chan))
        print('    Options:', enum_mask_to_string(OptionFlags, options))
        print('    Sample interval:  ' + str(sample_interval) + ' seconds')
        try:
            input("\nPress 'Enter' to continue")
        except (NameError, SyntaxError):
            pass

        print('\nAcquiring data ... Press Ctrl-C to abort')


        try:
if os.path.exists(basepath):
```

```
if not (os.path.exists(mypath)):
os.mkdir(mypath)
else:
os.mkdir(basepath)
os.chdir(basepath)
os.mkdir(mypath)
      except OSError as exc:
raise

      os.chdir(mypath)
      fileDateTime = datetime.strftime(datetime.now(), "%Y_%B_%d_%H%M%S")
      dirpath = os.getcwd()
      fileDateTime = mypath + "/" + fileDateTime + ".csv"
      csvfile = open(fileDateTime, "w+")
      csvwriter = csv.writer(csvfile)
      myArrayHeader = []
      myArrayHeader.append([])
      myArrayHeader[0].append('     Date/Time     ')

      # Display the header row for the data table.
      print('\n  Samples/Channel', end='')
      for chan in range(low_chan, high_chan + 1):
         print('     Channel', chan, end='')
         myArrayHeader[0].append('   Chan ' + str(chan))
      print('')
      csvwriter.writerows(myArrayHeader) # Write the array to file
      csvfile.flush()


      try:
         samples_per_channel = 0
         while True:
new_index = 0
myArray=[] #create an empty array
            # Display the updated samples per channel count
samples_per_channel += 1
print('\r{:17}'.format(samples_per_channel), end='')
myArray.append([])  #add a row to the array
# Time stamp:
timestamp = datetime.strftime(datetime.now(), "%Y %B %d %H:%M:%S.%f")
myArray[new_index].append(timestamp)
```

```
            # Read a single value from each selected channel.
for chan in range(low_chan, high_chan + 1):
value = hat.a_in_read(chan, options)
print('{:12.5} V'.format(value), end='')
myArray[new_index].append('{:12.5f}'.format(value))

stdout.flush()
csvfile = open(fileDateTime, "a")
csvwriter = csv.writer(csvfile)
csvwriter.writerows(myArray) # Write the array to file
csvfile.flush()

            # Wait the specified interval between reads.
sleep(sample_interval)

        except KeyboardInterrupt:
            # Clear the '^C' from the display.
            print(CURSOR_BACK_2, ERASE_TO_END_OF_LINE, '\n')

    except (HatError, ValueError) as error:
        print('\n', error)
csvfile.close()

if __name__ == '__main__':
    # This will only be run when the module is called directly.
    main()
```