Here's an example app that logs software polled data to a csv file. The data is collected using a timer (default is 1 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 134 Functions Demonstrated:
        mcc134.t_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 mcc134, HatIDs, HatError, TcTypes
from daqhats_utils import select_hat_device, tc_type_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.
    """
    tc_type = TcTypes.TYPE_J   # change this to the desired
thermocouple type
    delay_between_reads = 1  # In Seconds
    channels = (0, 1, 2, 3)
    basepath = '/home/pi/Documents/Measurement_Computing'
    mypath = basepath + '/log_files'

    try:
        # Get an instance of the selected hat device object.
        address = select_hat_device(HatIDs.MCC_134)
        hat = mcc134(address)

        for channel in channels:
            hat.tc_type_write(channel, tc_type)

        print('\nMCC 134 single data value read example')
        print('    Function demonstrated: mcc134.t_in_read')
        print('    Channels: ' + ', '.join(str(channel) for channel in
channels))
        print('    Thermocouple type: ' + tc_type_to_string(tc_type))
        print('    Sample rate:  ' + str(delay_between_reads) + '
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")
        fileDateTime = dirpath + "/" + 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  Sample', end='')
        for channel in channels:
            print('     Channel', channel, end='')
            myArrayHeader[0].append('   Chan ' + str(channel) + ' (C)')
        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{:8d}'.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")
myArray[new_index].append(timestamp)
                # Read a single value from each selected channel.
for channel in channels:
value = hat.t_in_read(channel)
if value == mcc134.OPEN_TC_VALUE:
print('     Open     ', end='')
```

```
#append a num_channels of data to the array (ROW)
myArray[new_index].append('     Open     ')
elif value == mcc134.OVERRANGE_TC_VALUE:
print('     OverRange', end='')
myArray[new_index].append('     OverRange')
elif value == mcc134.COMMON_MODE_TC_VALUE:
print('   Common Mode', end='')
myArray[new_index].append('     Common Mode')
else:
print('{:12.2f} C'.format(value), end='')
myArray[new_index].append('{:12.4f}'.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(delay_between_reads)

        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()
```