Preliminary Documentation for pyTES

Intro

pyTES is a software package meant to try and analyze various UCB TES data, from Lakeshore and SQUIDs. The SQUID-TES is currently unavailable, however the Lakeshore-TES is available.

There are several files involved in pyTES:Lakeshore and dependencies:

Dependancies

- python 3
- ROOT with pyROOT
- numpy
- scipy
- matplotlib
- pandas
- readROOT
  - A function using pyROOT to read ROOT files easily
- cloneROOT
  - A function using pyROOT to clone and write a ROOT file easily

Base functions used

- vcl2root.py
  - This method takes a VCL file and can convert it to a csv and/or a ROOT file. The csv file output is what you *should* get from the Triton software (if it worked...)

```
usage: vcl2root.py [-h] [-f INPUTVCLFILE] [-o OUTPUTFILE] [-t OUTPUTTYPE]

optional arguments: -h, --help show this help message and exit

-f INPUTVCLFILE, --inputVCLFile INPUTVCLFILE Specify the full path of the VCL file you wish to convert

-o OUTPUTFILE, --outputFile OUTPUTFILE Specify output file. If not a full path, it will be output in the same directory as the input file

-t OUTPUTTYPE, --outputType OUTPUTTYPE Specify the output file type, csv (default) or root
```

```
python3 vcl2root.py -f ~/cuore/bolord/run8/log_170922_171648.vcl -o run8 -t root
```

- merge_fridge_bridge.py
  - This function merges the fridge log file, in root form, with data from the Lakeshore DAQ that records TES data. It takes in the bridge data as well as a fridge file, and then puts them into the specified output file.
  - Since time sampling is not uniform between the LSB and the fridge, the fridge data is interpolated to the timestamps specified by the bridge data (since those are actual observation times of TES data).

> usage: merge_fridge_bridge.py [-h] [-b INPUTBRIDGEFILE] [-f INPUTFRIDGEFILE] [-o OUTPUTFILE] [-i INTERPTYPE] [-d INTERPDEGREE]
>
> optional arguments: -h, --help show this help message and exit
>
> -b INPUTBRIDGEFILE, --inputBridgeFile INPUTBRIDGEFILE Specify the full path to the bridge root file you wish to merge with fridge data
>
> -f INPUTFRIDGEFILE, --inputFridgeFile INPUTFRIDGEFILE Specify the full path of the Fridge root file you wish to merge with bridge data
>
> -o OUTPUTFILE, --outputFile OUTPUTFILE Specify output root file. If not a full path, it will be output in the same directory as the input bridge file
>
> -i INTERPTYPE, --interpType INTERPTYPE Specify interpolation method: "interp" for standard interpolation and "spline" for interpolated univariate spline. Default is "spline"
>
> -d INTERPDEGREE, --interpDegree INTERPDEGREE Specify the interpolation degree (default 3 for cubic)

```
python3 merge_fridge_bridge.py -b ~/cuore/bolord/run8/Scanner_20170925_103058_p00000.
root -f ~/cuore/bolord/run8/run8.root -o run8_epcal_ch2.root
```

- tmf2root.py
  - This function will read in a Noise Thermometer tmf file and can output it into a specified root file, appending if needed.

> usage: tmf2root.py [-h] [-r INPUTROOTFILE] [-n INPUTNTFILE] [-o OUTPUTFILE] [-i INTERPTYPE] [-d INTERPDEGREE]
>
> optional arguments: -h, --help show this help message and exit
>
> -r INPUTROOTFILE, --inputROOTFile INPUTROOTFILE Specify the full path of the ROOT file you wish

> to inject NT data into
>
> -n INPUTNTFILE, --inputNTFile INPUTNTFILE Specify the full path of the Noise Thermometer file you wish to use
>
> -o OUTPUTFILE, --outputFile OUTPUTFILE Specify output root file. If not a full path, it will be output in the same directory as the input file
>
> -i INTERPTYPE, --interpType INTERPTYPE Specify interpolation method: "interp" for standard interpolation and "spline" for interpolated univariate spline. Default is "spline"
>
> -d INTERPDEGREE, --interpDegree INTERPDEGREE Specify the interpolation degree (default 3 for cubic)

```
python3 tmf2root.py -r ~/cuore/bolord/run8/run8_epcal_ch2.root -n ~/cuore/bolord/run8
/2017-09-25_10-29-50.tmf -o ch2_3.16uA.root
```

- tc_stuff.py
  - This is the final script that will output plots
  - Typically one specifies the curve type and the fitting method, the input excitation current (in A), the channel and whether it is the NT or EP used for temperature fits.
  - Output is by default sent to a directory in the basepath of the input root file.
  - You can also output text files using the -d option
  - Error propagation is not yet satisfactory.
  - It is for now strongly recommended to use `tanh` and `curve` for the fits.
  - A very experimental fit option via `-m odr` can use scipy orthogonal distance regression but it requires proper x and y errors.
  - Plots should be contained in the output directory and will contain enough information in the name to allow, for a given channel, both the EP and NT fits if desired.

> usage: tc_stuff.py [-h] [-f FILEPATH] [-c CHANNEL] [-i CURRENT] [-m FITMETHOD] [-R FUNCTION] [-o OUTPUTPATH] [-d] [-s] [-v VOLTAGE] [-T TEMP]
>
> optional arguments: -h, --help show this help message and exit
>
> -f FILEPATH, --filePath FILEPATH Specify where to find data
>
> -c CHANNEL, --channel CHANNEL Specify which channel to run over, or if set to all, run over all channels
>
> -i CURRENT, --current CURRENT Specify which excitation current to fit. Enter value in units of A. Default is all. To select all currents enter all

> -m FITMETHOD, --fitMethod FITMETHOD Select fit method: SLSQP, Nelder-Mead, or curve for scipy curve_fit (default)
>
> -R FUNCTION, --function FUNCTION Select the functional form to fit: erf or tanh (default)
>
> -o OUTPUTPATH, --outputPath OUTPUTPATH Specify the output path to put plots. If not specified will create a directory in the input path based off the input filename. If not absolute path, will be relative to the input file path
>
> -d, --dumpFile Specify whether a dump file to store R and T in text should be made. Default location will be set to outputPath
>
> -s, --smooth Specify whether to use smoothing to handle overlap cases -- EXPERIMENTAL
>
> -v VOLTAGE, --voltage VOLTAGE Specify which excitation voltage to fit. Enter value in units of V. Default is all. To select all voltages enter all -- EXPERIMENTAL DO NOT USE
>
> -T TEMP, --temp TEMP Select the thermometer to use: nt for noise thermometer (default) or ep for ep cal

```
python3 tc_stuff.py -f ~/cuore/bolord/run8/ch2_3.16uA.root -R tanh -m curve -i 3.16e-6 -c 2 -T ep
```

Steps

1. Get all log files collected
2. Convert VCL to ROOT
3. Merge Fridge and Bridge data into common ROOT file
4. Parse and merge Noise Thermometer data into the common fridge and bridge ROOT file
5. Generate Tc fits

Note that if you do not use fridge data and/or NT data, `tc_stuff` is not yet robust enough to not care (though it could be made so pretty easily actually...sorry).

Helper bash functions

Below are a list of helpful bash functions you can put in your .bashrc to make getting data files relatively easy. You will need to change them to suit you since they require ssh to moller and mott.

Example:

To get all VCL files that match the wildcard search `*1803*` in their filename:

```
gvcl 1803 ~/cuore/bolord/run12
```

This will attempt to then connect to cabibbo through moller and grab all log files whose name contains `1803` and send them to the specified path. If no path is specified they will go to the current directory.

Run `spaceToUnderscore` to convert all file names with spaces in the current directory to files with underscores instead of spaces. THIS IS NOT UNDOABLE!

All bash functions:

```
#    gvcl    get vcl file
#    ---------------------------------------------------
    gvcl() {
        if [ ! -n "$1" ]; then
            echo "Enter a log file string to search for"
        else
            if [ ! -n "$2" ]; then
                place=./
            else
                place=$2
            fi
            # Create tunnel through moller to scp port but make it close after 10 sec
onds of no activity
            # this forwards port N from localhost to proxy and on the proxy says any
activity on N gets
            # forwarded through port M on the target: -LN:<target>:M proxy
            ssh -f -L12345:169.229.38.141:22 bwelliver@moller sleep 10
            scp -P12345 Administrator@localhost:/home/Administrator/Data/*$1*.vcl $pl
ace
        fi
    }

#    gvcl    get NT file
#    ---------------------------------------------------
    getNT() {
        if [ ! -n "$1" ]; then
            echo "Enter a NT file string to search for"
        else
            if [ ! -n "$2" ]; then
                place=./
            else
                place=$2
            fi
            # Create tunnel through moller to scp port but make it close after 10 sec
onds of no activity
            # this forwards port N from localhost to proxy and on the proxy says any
activity on N gets
            # forwarded through port M on the target: -LN:<target>:M proxy
            ssh -f -L12346:169.229.38.130:22 bwelliver@moller sleep 10
            scp -P12346 admin@localhost:/cygdrive/c/MagniconData/TempViewerData/*$1*.
tmf $place
        fi
    }

#   getBridge   get a Scanner file(s) based on generic pattern
#   ----------------------------------------------------------------------------
    getBridge() {
        if [ ! -n "$1" ]; then
            echo "Enter a scanner file string to search for"
        else
            if [ ! -n "$2" ]; then
                place=./
            else
                place=$2
            fi
            echo "rsync -avz --progress bolord@mott:~/Scanner*$1*.root $place"
            rsync -avz --progress bolord@mott:~/Scanner*$1*.root $place
        fi
    }

#    spaceToUnderscore    in current directory replace spaces in file names with unde
rscores in file_names
#    --------------------------------------------------------------------------------
--
    spaceToUnderscore() {
        for f in *\ *; do mv "$f" "${f// /_}"; done
    }
```