OPENQCM

Name

openQCM Next GUI Software

Programming language

Python

Version

0.1.2

Date

2020-07-22

Author

openQCM Team - Marco (mainly)

Supporting and powering the openQCM project

Novaetech S.r.l

Description

The opeQCM NEXT software application is developed in Python,
which is an open source, object - oriented and suited for scientific application programming language.
Python makes the software program easy to modify and develop for custom application.

The new opeQCM NEXT software is able to exploit all the main functionalities of the device. Real time monitoring of frequency and dissipation on the fundamental mode and overtone harmonics. It is possible to acquire almost simultaneously 5 sweep signals and elaborate the frequency and dissipation measurement in roughly 700 msec. In addition, the application allows to control and monitor the sensor module temperature in real time.

The application is based mainly on multiprocessing package (https://docs.python.org/3/library/multiprocessing.html).

History Changes

version 0.1.2

Python Software Major Changes

- Major refactor of the software GUI using QT designer. Main changes maximize - minimize the GUI window
    - [SOLVED] bug causing the GUI to be incorrectly displayed on some laptops

- [SOLVED] bug in mainWindow.py causing the freeze of the software when interacting with the resizing GUI window

in mainWindow.py

```
def updateViews1():
    self._plt0.clear()
    if self._get_source() != SourceType.multiscan:
        self._plt1.clear()
```

- Switch ON/OFF temperature control in measurement mode for multiscan and single measurement
  added another Boolean entry in the config.ini file
- BUG: dissipation calculation fails when the sweep signal goes under zero level
  ISSUE: Skewness of left-side of the gain resonance curve
  in Multiscan.py and Serial.py

  [SOLVED] Developed a new algorithm for the calculation of dissipation:
  - shift the signal up make the gain resonance curve positive
  - select the bandwidth at 50% threshold of the maximum gain, considering only the right-side of the resonance curve
  - multiply x2 the bandwidth

  ```
  # select the bandwith at 50% threshold of the maximum gain, considering
  only the right-side of the resonance curve
  # multiply x2 the bandwith
  bandwidth = (freq[index_M] - freq[i_max])*2
  ```

- change the sweep range in constants.py same sweep range for all overtones
- Set y-axis range of frequency and dissipation to optimize the view in multiscan and single mode
  in mainWindow.py example routine used in multiscan and single mode

```
# VER 0.1.2
# get y_freq and y_dissipation max value
y_freq_max = np.nanmax(y_freq)
y_freq_min = np.nanmin(y_freq)
y_diss_max = np.nanmax(y_diss)
y_diss_min = np.nanmin(y_diss)
# print ("GET MAXIMUM VALUES FREQ AND DISSIP = ")
# print (y_freq_max), print (y_diss_max)

# VER 0.1.2
# set the y-range of dissipation and frequency axis
try:
    self._plt2.setYRange(y_freq_min - 100, y_freq_max + 100, padding = 0)
    self._pltD.setYRange(y_diss_min - 0.000001, y_diss_max + 0.000001,
padding = 0)
except:
    pass
```

```
  in multiscan mode is necessary to init a y-range lists

  ```python
  # multiscan y-range limit lists
  self._y_freq_max = [0, 0, 0, 0, 0]
  self._y_freq_min = [0, 0, 0, 0, 0]
self._y_diss_max = [0, 0, 0, 0, 0]
  self._y_diss_min = [0, 0, 0, 0, 0]
```

- [SOLVED] Bug: first line in datalog file missing frequency and dissipation data array
  in worker.py

```
# VER 0.1.2
if (self._overtone_number == index_store) and (self._time_store[index_store]
- self._timestart)/_millisec > 0):
```

- [SOLVED] Bug: worker processes are still running when you press stop button on main gui
  when stop is called, terminate the processes alive in the worker

```
# VER 0.1.2
if self._acquisition_process is not None and
self._acquisition_process.is_alive():
        # stop the process
        self._acquisition_process.stop()
        # self._acquisition_process.join(Constants.process_join_timeout_ms)

        # terminate the process
        #
https://docs.python.org/3/library/multiprocessing.html#multiprocessing.Proce
ss.terminate
        self._acquisition_process.terminate()
        # wait for a while
        sleep(1)
```

- [SOLVED] BUG Pressing the PID set button when measurement mode is NOT running,
  caused the freeze of the software. Add a short delay of 0.100 sec between each serial write
  command.
  in mainWindow.py

```
  def PID_Set (self):
      ...
      # set PID parameters
        cycling_time_msg = 'C' + str(int(_var_cycling_time)) + '\n'
        # VER 0.1.2 add a short sleep for communication
        sleep(0.1)
        self._my_serial.write(cycling_time_msg.encode())
        sleep(0.1)
        ...
```

- [SOLVED] BUG: datalog file name is the same, causing append of different measurement
  sessions in the same file. log a new data file each time the start button is pressed, data log
  file name set at the start of the measurement session
  in worker.py
```

```
    # VER 0.1.2
    # init the new datalog file in single mode
    filenameCSV = "{}_{}".format(self._csv_filename, self._overtone_name)

    # VER 0.1.2
    # init the new datalog file in single mode
    filenameCSV = "{}_{}".format(self._csv_filename, "multi_")
```

Python Software Minor Changes

- [SOLVED] BUG in fileStorage.py insert the header in datalog file
- [SOLVED] BUG first line in datalog file missing frequency and dissipation data array in worker.py

```
# debug version 0.1.2
if (self._overtone_number == index_store) and (self._time_store[index_store]
- self._timestart)/_millisec > 0):
```

- 5 MHz default option in calibration mode file in Calibration.py

```
@staticmethod
def get_speeds():
    #:return: List of the common baud rates, in bps :rtype: str list.
    # return [str(v) for v in ['@10MHz_QCM', '@5MHz_QCM']]#[1200, 2400,
4800, 9600, 19200, 38400, 57600, 115200]]
    # VER 0.1.2
    return [str(v) for v in ['@5MHz_QCM', '@10MHz_QCM']]
```

- Clear all plot when start button is pressed. Added clear of amplitude and phase sweep plot in mainWindow.py

```
def clear(self):
    support = self.worker.get_d1_buffer()
    if support.any:
        if str(support[0])!='nan':
            print(TAG, "All Plots Cleared!", end='\r')
            self._update_sample_size()
            self._plt2.clear()
            self._pltD.clear()
            self._plt4.clear()

            # VER 0.1.2
            # clear amplitude and phase sweep plot
            self._plt0.clear()
            self._plt1.clear()
```

- Optimize and update infobar and infostatus in multiscan mode

```
if vector1.any:
    # progressbar
    if self._ser_control <= Constants.environment:
        self._completed = self._ser_control * 2
```

```
        # VER 0.1.2a
        # Optimize and update infobar and infostatus in multiscan mode
        labelstatus = 'Processing'
        color_err = '#000000'
        labelbar = 'Please wait, processing early data'
        self.ui.infostatus.setText("<font color=#000000> Program Status
</font>" + labelstatus)
        self.ui.infobar.setText("<font color=#0000ff> Infobar </font><font
color={}>{}</font>".format(color_err,labelbar))

    # progressbar
    self.ui.progressBar.setValue(self._completed + 2)

    if self._ser_control == Constants.environment:
        # clear plt reset buffer at the end of processing early data
        self.clear()

        # VER 0.1.2a
        # Optimize and update infobar and infostatus in multiscan mode
        labelstatus = 'Monitoring'
        color_err = '#000000'
        labelbar = 'Monitoring!'
        self.ui.infostatus.setText("<font color=#000000> Program Status
</font>" + labelstatus)
        self.ui.infobar.setText("<font color=#0000ff> Infobar </font><font
color={}>{}</font>".format(color_err,labelbar))
```

- Changed the temperature Y minimum range to 5° C, in mainWindow.py

```
# VER 0.1.2a
# change the Temperature Y-range to 5 - 45 °C
self._plt4.setYRange(5, 45, padding = 0)
```

- Add colour to frequency and dissipation label to enhance the visibility of data plot
- Changed set reference button and added reset reference button for optimization in mainWindow.py

```
# VER 0.1.2
# changed the function set reference in single mode

# set flag reference true
self._reference_flag = True
    [...]

# VER 0.1.2
# add reset reference button / function
def reference_not (self):
    [...]
```

- Editing pyqtgraph context menu to get rid of 'Plot Options' and 'Export' in mainWindow.py example code related to frequency real time plot

```
# VER 0.1.2
# editing pyqtgraph context menu
# https://groups.google.com/g/pyqtgraph/c/h-dyrOl6yZU/m/NpMQxh-jf5cJ
# get rid of 'Plot Options'
self._plt2.ctrlMenu = None
# get rid of 'Export'
self._plt2.scene().contextMenu = None
```

- Added the capability to switch between PyQt5 and PySide2
  in app.py

  ```
  # VER 0.1.2
  # importing from PyQt5 or PySide2
  try:
      from PyQt5 import QtGui
  except:
      from PySide2 import QtGui
  ```

  the application is executed in different way

  ```
  # VER 0.1.2
  # execute the application in PyQt or PySide style
  if 'PyQt5' in sys.modules:
      self._app.exec()
  else:
      self._app.exec_()
  ```

- [SOLVED] bug using macOS Big Sur, Architecture.get_os() is not able to get the correct OS

  ```
  # VER 0.1.2
  # set directory slash, solving bug for macOS Big Sur
  # sets the slash depending on the OS types
  if Architecture.get_os() is (OSType.linux or OSType.macosx):
      # print ("MAC_OS_X")
      slash = "/"

  elif Architecture.get_os() is OSType.windows:
      # print("WINDOWS")
      slash = "\\"
  else:
      # print ("OTHER_OS")
      slash = "/"
  ```

---

version 0.1.1c

Python Software Major Changes

- MainWindow.py added a drop-down menu to recall PID parameters setting factory default and openQCM tested parameters
- Multiscan.py and Serial.py added a sleep for the transmission of the PID parameter

- Multiscan.py and Serial.py moved self._Temperature_PID_control() in a try - except block code

```
try:
    # SET TEMPERATURE and PID PARAMETERS
    # ------------------------------------------------
      self._Temperature_PID_control()
  except:
    print(TAG, "EXCEPTION: set temperature control failed")
    # Log.i(TAG, "EXCEPTION: exception at serial port read process")
    self._flag_error_usb = 1
```

- Multiscan.py and Serial.py insert a timeout in the while acquisition loop to prevent infinite loop 500 msec

```
_time_elapsed = time() - timeStart
    ...
  if  _time_elapsed > Constants.TIME_ELAPSED_TIMEOUT
    ...
```

- Multiscan.py and Serial.py added a short sleep of 100 msec after serial write

```
  # added a short sleep before read serial
  sleep(Constants.WRITE_SERIAL_WAIT)
```

- Multiscan.py and Serial.py check the length of the serial read buffer if exceed the number of samples = 500

```
if length > Constants.argument_default_samples + 2:
    ...
```

- Multiscan.py and Serial.py Reset buffer if an exception at serial port occurs

```
  # reset buffer
  data_mag = np.linspace(0,0,samples)
  data_ph  = np.linspace(0,0,samples)
```

Python Software Minor Changes

- mainWindow.py _set_PID_T_default value of temperature changed to 25000 as default
- mainWindow.py change the format of the config file in decimal using fmt='%d' https://numpy.org/doc/stable/reference/generated/numpy.savetxt.html

```
np.savetxt( _path,  np.row_stack( [data0, data1, ... dataN] ), fmt='%d')
```

version 0.1.1a

Python Software Major Changes

- Added frequency and dissipation current values indicator, both in multiscan and single mode
- Added overtone data selector, in multiscan mode
- Modified logged data in multiscan mode.
  Solved bug: data were written for each new harmonic sweep, resulting in duplicate
- Solved a bug for PID set parameters, causing the sw freeze. Reason incompatible config.txt file format
- Added reset to default Temperature and PID value on config.ini and GUI indicator when: i) STOP measurement and ii) software initialization
- Improved the multi harmonic sweep amplitude graph, by passing through the processes a queue made of each individual sweep curve

Python Software Minor Changes

- Add the openQCM icon in taskbar and application
  windows workaround in taskbar solution reported here
  https://stackoverflow.com/a/12522799/4030282
- version 0.1.1a logged data file: changed the format of dissipation in "no format"
- version 0.1.1a sweep curves bug fixed for 5 MHz quartz resonators

Intended Audience

Science/Research/Engineering

Software Development

User Interfaces

Requirements

Requirements:

- Python 3.7 (verified compatibility with Python 3.6) (https://www.python.org/).
- Anaconda3-5.3.0
  External Packages:
  - PyQt5 5.9.2 (https://pypi.org/project/PyQt5/).
  - PySerial 3.4 (https://pypi.org/project/pyserial/).
  - PyQtGraph 0.10.0 (http://www.pyqtgraph.org/).

Other internal packages used:

- multiprocessing, numpy, scipy, setuptools, io, platform, sys, enum, argparse, cvs, time, datetime, logging, etc.

Installation instructions/guide:

Download openQCM Next Python source code version 0.1.1a here:
https://openqcm.com/shared/next/software/openQCM_Next_py_0.1.2_source.zip

Windows, macOS

1. Download and install Anaconda3 for Python 3.7 version Anaconda3-5.3.0 https://www.anaconda.com/download/
2. Open Anaconda3 prompt (Windows) or terminal (macOS) and type (install/upgrade Python packages) :

conda install pyqtgraph pyserial

Linux

1. Type the command below by replacing username with that of your pc change permission of Anaconda3 sudo chown -R username:username /home/username/anaconda3

2. Open Anaconda3 terminal and type (install/upgrade Python packages) :
   conda install pyqtgraph pyserial
3. Set permission on serial port
   sudo usermod -a -G uucp username
   sudo usermod -a -G dialout username
4. Logout and Login

Usage

Start the application from Anaconda3 prompt

1. Launch Anaconda3 prompt
2. Browse to the openQCM Q-1 Python software main directory
   ...\openQCM_Next_py_0.1.2_source\OPENQCM\openQCM
3. launch the python application main GUI by typing the command
   python –m openQCM

Start the standalone executable application developed for windows OS 64-bit sta

1. Donwload the .zip compressed application files here:
   https://openqcm.com/shared/next/software/openQCM_Next_py_0.1.2_exe.zip
2. Unzip the package and browse to directory:
   .../openQCM_Next_py_0.1.2.exe
3. Launch the app.exe application shortcut

Directory structure:

```
<DIR>          app
               app.exe.lnk
<DIR>          logged_data
<DIR>          openQCM
                     Calibration_5MHz.txt
                     Calibration_10MHz.txt
                     config.txt
                     PeakFrequencies.txt
```

Links

- [website] https://openqcm.com/
- [github] https://github.com/openQCM

Contact

- [mail] info@openqcm.com

License and Citations

The project is distributed under GNU GPLv3 (General Public License).