vtipy Documentation

Release 0.1

Jack Hodkinson

Nov 19, 2018

CONTENTS

CHAPTER

ONE

VTIPY PACKAGE

1.1 vtipy.impedance module

class vtipy.impedance.**solartron1260**(*Vac=0.5*, *Vdc=0.0*, *integration_time=1*, *gpib_address=2*)

Bases: `Gpib.Gpib.Gpib`

Parameters

- **Vac** (*float, optional*) – A.C. voltage for impedance measurements.
- **Vdc** (*float, optional*) – D.C. bias.
- **integration_time** (*int, optional*) – Integration time in seconds.
- **gpib_address** (*int, optional,*) – Gpib address of the instrument. Default is 2.

Example

Here is a demonstration of how the object is initialised and how a frequency dependent measurement is executed.

```
>>> from vtipy import solartron1260
>>> sol = solartron1260()
>>> sol.measure_impedance( scan_number = 1 )
```

measure_frequency(*frequency*)

Sends an instruction to the Solarton to measures impedance at the specified frequency.

Parameters **frequency** (*float*) – frequency for measurement.

Returns **result** – Impedance data formatted as a comma separated string. The first three values (frequency, magnitude, and the argument) are the important values that need to be saved.

Return type string

measure_impedance(*scan_number*, *filename=None*, *fmin=0.05*, *fmax=9800000.0*, *ppd=20*, *Tset=None*, *Tcell=None*, *scan_label=None*)

Sends an instructions to the Solartron to make an impedance measurement over a range of frequencies. This is accomplished by executing the solartron1260.measure_impedance_process method as a subprocess as to allow a main control script to continue to monitor temperature while this runs in parallel.

Parameters

- **scan_number** (*int, optional*) – This should be specified as to indicate the relative order of the measurement.

- **filename** (*None, string*) – The name of file in which impedance data will be stored. If left as None it will be assigned a name based on the scan number: "scan_n.txt", where n = scan_number.
- **fmin** (*float, optional*) – The minimum frequency of the impedance sweep. The default value is 0.05.
- **fmax** (*float, optional*) – The maximum frequency of the impedance sweep. The default value is 9.8e6.
- **ppd** (*int, optional*) – The number of points per decade at which an impedance measurement will be recorded. Default is 20.
- **Tset** (*None, int*) – The setpoint temperature from the Eurotherm. If set it is stored in the header of the datafile. It should be set during variable temperature experiments. If left as none a NAN value will be recorded in the header.
- **Tcell** (*None, int*) – The "cell temperature" as measured by the MAX31855 thermocouple. If set it is stored in the header of the datafile. If left as none a NAN value will be recorded in the header.
- **scan_label** (*None, string*) – Optional label to provide extra meta-data for later analysis. Typically this is set to "up" or "down" to distinguish measurements during an upward or downward temperature sweep. This value is recorded in the header of the datafile.

measure_impedance_process (*scan_number*, *filename*, *fmin*, *fmax*, *ppd*, *Tset*, *Tcell*, *scan_label*)

Instructs the Solartron to measure the impedance over a range of frequencies.

This method is designed to be run in parallel with a main script as to allow the main script to continue to monitor process temperatures while the impedance is measured. This is accomplished by using the solartron1260.measure_impedance method which calls this method (the solartron1260.measure_impedance_process) as a sub-process. However, this method may be run independently if no other measurements are required to be run in parallel.

The parameters of the method are the same as those detailed in the solartron1260.measure_impedance method, however, the default values set for that method are not set here. They all must be provided explicitly.

Parameters

- **scan_number** (*int*) – This should be specified as to indicate the relative order of the measurement.
- **filename** (*None, string*) – The name of file in which impedance data will be stored. If set as None it will be assigned a name based on the scan number: "scan_n.txt", where n = scan_number.
- **fmin** (*float*) – The minimum frequency (in Hz) of the impedance sweep. Cation should be taken when setting this value to anything less than 0.05 Hz as errors with the Gpib.Gpib library are encountered.
- **fmax** (*float*) – The maximum frequency (in Hz) of the impedance sweep. Max value is 1e7.
- **ppd** (*int*) – The number of points per decade at which an impedance measurement will be recorded.
- **Tset** (*int*) – The setpoint temperature from the Eurotherm. The value is stored in the header of the datafile.

- **Tcell** (*int*) – The "cell temperature" as measured by the MAX31855 thermocouple. The value is stored in the header of the datafile.
- **scan_label** (*string*) – A label to provide extra meta-data for later analysis. Typically this is set to "up" or "down" to distinguish measurements during an upward or downward temperature sweep. This value is recorded in the header of the datafile.

send(*msg*)

Sends a message to the solartron. Just like the Gpib.Gpib.write except a 0.02 s sleep time is applied to prevent too many signals being sent at the same time.

Parameters **msg** (*string*) – The message to send the solartron. This should be one of the instructions from the instruction set found in the solartron manual.

Returns

Return type None

1.2 vtipy.temperature module

This module is designed to control the temperature controllers used in the variable temperature impedance control system.

class vtipy.temperature.**temperature_controllers**(*filename*, *serial_usb_port*, *addr=1*, *CLK=25*, *CS=24*, *DO=18*)

Bases: `minimalmodbus.Instrument`, `Adafruit_MAX31855.MAX31855.MAX31855`

This class controls both the MAX31855 temperature controller and the Eurotherm 3216.

Methods are defined to allow measurement and recording of the temperature from the Eurotherm and the MAX31855 as well as to change the setpoint tempreature of the Eurotherm controller.

This control is accomplished by inheritance from the minimalmodbus.Instrument and MAX31855.MAX31855 classes.

Parameters

- **filename** (*string*) – The name assigned to the file in which temperature measurements will be recorded.
- **serial_usb_port** (*string*) – The path associated with the serial-to-USB converter on the Raspberry Pi filesystem.
- **addr** (*int, optional*) – The slave address of the serial-to-USB converter (connected to the Eurotherm controller).
- **CLK** (*int, optional*) – I/O port on the Raspberry Pi associated with the clock pin on the MAX31855.
- **CS** (*int, optional*) – I/O port on the Raspberry Pi associated with the "chip sellect" port on the MAX31855.
- **DO** (*int, optional*) – I/O port on the Raspberry Pi associated with the "data out" port on the MAX31855.

Example

The following is an example of how to initialise the temeprature controller object, make an initial temperature measurement, ramp the temperature, and hold the temperature at a desired setpoint.

```
>>> from vtipy import temperature_controllers
>>> tc = temperature_controllers("temp.txt", serial_usb_port)
>>> Tcell = tc.measure_temperatures()
>>> tc.ramp_temperature( Tset = 100 )
>>> tc.hold( hold_time = 60)
```

`hold`(*hold_time*, *temp_resolution=2*)

This function holds the setpoint temperature of the Eurotherm while measuring and recording the process temperatures at the specified temperature resolution.

Parameters

- **`hold_time`** (*`int`*) – Time in minutes to hold the Eurotherm setpoint.
- **`temp_resolution`** (*`int`*) – Temperature measurements per minute.

`measure_temperatures`()

This method serves two purposes: Firstly, it measures the two cell and furnace temperature. Secondly, this method records time since the initial temperature measurement (dt), the setpoint temperature from the Eurotherm.

The measurement is recorded into the temperature datafile specified when an instance of this class is defined. The data is added to the file as a comma separated line of the format:

” dt, Tsetpoint, Teuro, Tcell, Ttc_internal “

Returns **Tcell**

Return type string

`ramp_temperature`(*Tset*, *ramp_time=1*)

Parameters

- **`Tset`** (*`int`*) – Setpoint temperature in degrees C.
- **`ramp_time`** (*`int`*) – Time (in min) per degree C.

`read_process_temperature`()

Returns the furnace temperature from the Eurotherm

`read_setpoint`()

Reads the setpoint temperature from the Eurotherm. This is read from register 5 of the Eurotherm 3216. This might need to be adapted for other Eurotherm controllers. See the Eurotherm manual.

Returns The setpoint temperature from the Eurotherm as a string.

Return type setpoint (string?)

`set_setpoint`(*Tset*)

Sets the setpoint temperature of the Eurotherm. This is accomplished by setting register 24 of the Eurotherm 3216. This might need to be adapted for other Eurotherm controllers. See the Eurotherm manual.

Parameters **`Tset`** (*`int`*) – The new setpoint temperature (in degrees Celsius) for the Eurotherm controller.

PYTHON MODULE INDEX

INDEX