Python-based device control and automated measurements

Mirald Tuzi, José Soto, Carlos Gaspar Benitez Montiel, Justo Martín-Albo, Augusto Martins

Overview

31.05.2023

Overview

- Initially started as a small script for metalens measurements
  - Core ideas
    - Control all needed devices using only one Python script
    - Run measurement automatically
- This type of script can be generalised to take an arbitrary number of devices and measurement commands
- Creation of device-control tool
  - Python script aiming to enable arbitrary setup of measurement automation
  - Currently supported devices:
    - Arduino-controlled stepper motors (either through pyfirmata or pyserial, thanks to Carlos for programming Arduino)
    - Keithley picoammeter (either through pyvisa or pyserial, thanks to José for making compatible with pyserial)

Working principle

- Register all devices that will be used in a measurement run
- Define number of measurement samples
- Define a list of commands for measurement
- Run the measurement and store the output

- Code structure:
  - **User class**: manages the concrete measurement setup (useful if e.g. different measurement runs for the same setup are needed)
    - The user class inherits from the “measurements” class in which commands are processed
      - “measurements” inherits from “device_control” class through which one can communicate with all supported devices

Example

```python
class df_measurements(measurements.measurements):
    def __init__(self, my_init_commands={}, my_step_commands={}, my_reset_commands={}, nsamples=1):
        super().__init__(my_init_commands, my_step_commands, my_reset_commands, nsamples)

    def run_angular_df_measurement(self,savepath):

        self.register_keithley('/dev/cu.usbserial-110','pyserial')

        self.register_pyserial_motor('axial',   '/dev/cu.usbmodem11401', {'fw': b'1', 'bw': b'2'}, 10) #degrees
        self.register_pyserial_motor('angular', '/dev/cu.usbmodem11401', {'cw': b'3', 'ccw': b'4'}, 5) #

        self.nsamples = 3

        self.my_init_commands = {'argument': 'angle [°]'} # define as function of what the measurements will be performed

        self.my_step_commands = {'axial'               : (5, 'bw'),
                                 'measure current'     : 'I_no_df [nA]',
                                 'axial'               : (5, 'fw'),
                                 'angular'             : (1, 'cw'),
                                 'measure current'     : 'I_with_df [nA]'}

        self.my_reset_commands = {'angular': (self.nsamples, 'ccw')}

        self.connect_keithley()
        self.connect_steppers()

        if not self.keithley.connected:
            print("Error! Keithley not connected. Measurement not possible. Exiting...")
            return
        if self.stepper_motors=={}:
            print("Error! Steppers not defined. Measurement not possible. Exiting...")
            return

        self.run_measurement()

        self.df = pd.DataFrame(self.currents)
        self.df[self.argument] = [i*self.stepper_motors['angular'].res for i in range(self.nsamples)]
        #df['I_normed [nA]']     = df['I_with_df [nA]'].values / df['I_no_df [nA]'].values
        #df['I_normed_err [nA]'] = utils.gauss_reg_ratio(df['I_with_df [nA]'].values, df['I_no_df [nA]'].values)

        self.save_measurement(savepath)

    def run_some_other_df_measurement(self):
        pass
```

Register devices

Number of samples

Define measurement

Run and store measurement

```python
measurement = df_measurements()
savepath = '../../test'
measurement.run_angular_df_measurement(savepath)
```

```
> step 0
>>> Measurement sample taken!
> step 1
>>> Measurement sample taken!
> step 2
>>> Measurement sample taken!
> storing measurements under ../../test_2023-05-26_19.12.47.csv
> Measurement complete!
```

```python
pd.read_csv('../../test_2023-05-26_19.12.47.csv')
```

	I_no_df [nA]	I_no_df_err [nA]	angle [°]
0	2.267663e-14	2.113422e-14	0
1	2.127813e-14	1.929037e-14	5
2	2.390638e-14	2.688659e-14	10

Summary

- First working version of python script to control small setup and perform simple measurement

Outlook

- Optimise further
- Expand device support
  - José is already working on control of BGA and Monochromator via Python
- Integrate all of this into the graphical application that José has written so e.g. students can use it too