Experimental framework based on Sacred

The principle of Sacred is to store the experiment data and metadata in a file that can be stored in a database. Here we used the noSQL framework with MongoDB. There are numerous aspects to Sacred detailed in the original publication including:

1. Save code and imported code used to launch the experiment
2. View experiment parameter set
3. Save log
4. Draw graphs in the database and access the raw data (metrics)
5. Save numerous file types (artifacts)
6. Compare experiments: results/code difference/metadata difference

Install Sacred:

All the information about Sacred are here: visit Sacred

Install the database

Download MongoDB Create a new database called "sacred" (no caps).

Web interface

All the information about Omniboard are here: visit Omniboard)

in the command line type:

```
npm install -g omniboard
```

To launch omniboard: `` `omniboard -m 127.0.0.1:27017:sacred` ``

To open the interface connect to : http://localhost:9000/sacred

![](Images/2023-02-06-10-57-11.png)

Install the database quiery tool:

All the information about Incense are here: visit Incense)

Example of adaptation of [CSL-lights](XXX)

Script to control a light source	The same script as Sacred experiment

Script to control a light source

```
from serial import Serial
import CSLlight
arduino_port = "COM5"
sec = 1000 #conversion ms to s
LED_param = {'pin':11,
'offset':0.5*sec,
'period': 5*sec,
'duration': 2*sec,
'analog_value': 255,
}

link = Serial(arduino_port)
CSLLight.add_digital_pulse(link, LED_param)
CSLlight.start_measurement(link)
time.sleep(300)
CSLlight.stop_measurement(link)
```

The same script as Sacred experiment

```
from serial import Serial
import CSLlight
from sacred.observers import MongoObserver
from sacred import Experiment
ex = Experiment('blink_LED')
ex.observers.append(MongoObserver(db_name = "demo"))

@ex.config:
def cfg():
  arduino_port = "COM5"
  sec = 1000 #conversion ms to s
  LED_param = {'pin':11,
               'offset':0.5*sec,
               'period': 5*sec,
               'duration': 2*sec,
               'analog_value': 255,
               }
@ex.capture
def blink():
  link = Serial(arduino_port)
  CSLLight.add_digital_pulse(link, LED_param)
  CSLlight.start_measurement(link)
  time.sleep(300)
  CSLlight.stop_measurement(link)

@ex.automain
def run():
  blink()
```

Example of adaptation of [CSL-motors](XXX)

Script to control a motor

```
from CSLstage.CSLstage import CSLstage

arduino_port = "COM6"

stage = CSLstage(arduino_port, [1,1,1])
#gearbox ratio of X, Y and Z axis
stage.handle_enable(1)
stage.move_dx(10)
stage.handle_enable(0)
stage.link.close()
```

The same script as Sacred experiment

```
from serial import Serial
import CSLlight
from sacred.observers import MongoObserver
from sacred import Experiment
ex = Experiment('blink_LED')
ex.observers.append(MongoObserver(db_name = "demo"))

@ex.config:
def cfg():
  arduino_port = "COM5"
  gears = [1,1,1]

@ex.capture
def get_stage():
  stage = CSLstage(arduino_port, [1,1,1])

@ex.automain
def run():
  stage = get_stage()

  stage.handle_enable(1)    stage.move_dx(10)
  stage.handle_enable(0)
  stage.link.close()
```

Example of adaptation of [CSL-camera](XXX)

Script to control a camera	The same script as Sacred experiment

Script to control a camera

```
from CSLcamera import ControlCamera
cam_type = "MMConfig/Daheng.json"
update_param = {"Exposure": 150*1000,
              "Gain": 23}
downscale = 5 #downscale the image to save
cam = ControlCamera(cam_type, update_param, downscale)
N_im =  20
cam.snap_video(N_im)
video, timing = save_video("save_folder")
cam.reset()
```

The same script as Sacred experiment

```
from CSLcamera import ControlCamera

@ex.config
def config():
   cam_type = "MMConfig/Daheng.json"
   update_param = {"Exposure": 150*1000,
              "Gain": 23}

   downscale = 5 #downscale the image to save
   N_im =  20
@ex.capture
def get_camera():
   cam = ControlCamera(cam_type, update_param, downsc

@ex.automain
def run(N_im):
   cam.snap_video(N_im)
   video, timing = save_video(save_folder, _run)
```