



# Sardana Macros

by Zbigniew Reszela

(ALBA Synchrotron, Spain)

on behalf of Sardana Community



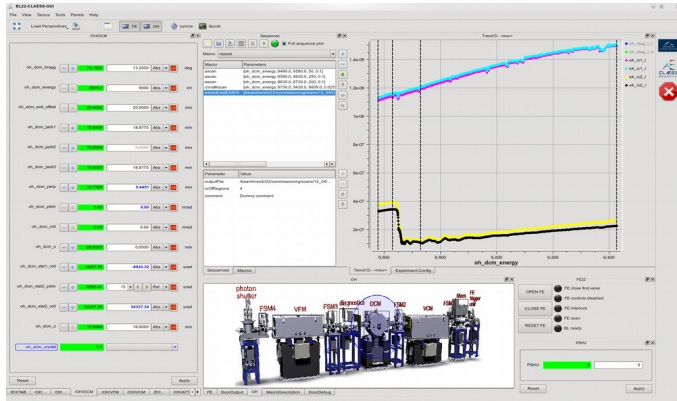


# Contents

- **Introduction**
  - What is Sardana and MacroServer
  - Usage of macros
  - Installation
- **Macro development**
  - Create simulated environment
  - Macro execution clients
  - Hello world example
  - Macro features
- **Scan with Tango attributes**



# Sardana is...



```

Door_zrezela_1 (13): lsmeas
Active Name Timer Eparis channels
-----
ng_odedtest oned01 oned01
* antgrp01 ct01 ct01, ct02, ct03, ct04
  antgrp02 ct01 ct01, ct02
  antgrp03 ct01 ct01, ct02, ct03, ct04, oned01

Door_zrezela_1 (14): lsm
Name Type Controller Axis
-----
gap01 PseudoMotor slitctrl01 1
icepap1302 Motor icepap13ctrl 2
mot01 Motor motctrl01 1
mot02 Motor motctrl01 2
mot03 Motor motctrl01 3
mot04 Motor motctrl01 4
mot05 Motor motctrl01 5
offset01 PseudoMotor slitctrl01 2
soprolec1 Motor soprolec_ctrl 1

Door_zrezela_1 (15): %scan mot01 0 1 4 0.1
Operation will be saved in /home/zrezela/tmp/test.h5 (w5)
Scan #329 started at Sun Oct 12 13:43:27 2014. It will take at least 0:00:00.694422
Moving to start positions...
#Pt No mot01 ct01 ct02 ct03 ct04 dt
0 0 0.1 0.2 0.3 0.4 0.085824
1 0.25 0.1 0.2 0.3 0.4 0.249444
2 0.5 0.1 0.2 0.3 0.4 0.410941
3 0.75 0.1 0.2 0.3 0.4 0.570931
4 1 0.1 0.2 0.3 0.4 0.730435
Operation saved in /home/zrezela/tmp/test.h5 (w5)
Scan #329 ended at Sun Oct 12 13:43:28 2014, taking 0:00:00.845093,Dead time 40.9%
(function dead time 29.9%)

Door_zrezela_1 (16):
  
```

Scientific SCADA Software Suite  
Suite = Sardana & Taurus projects

Spock – IPython based CLI

100 % Python

Built on top of TANGO

Extendable with plugins

Community driven:



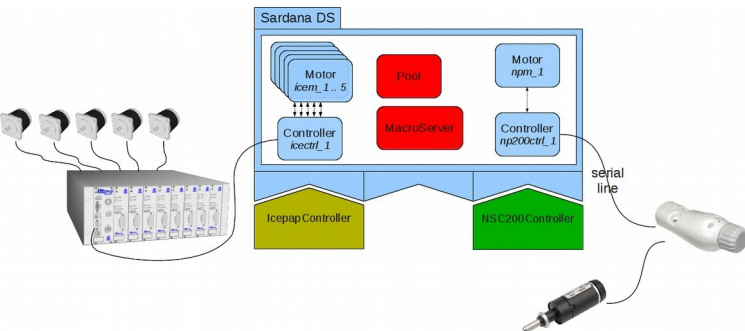
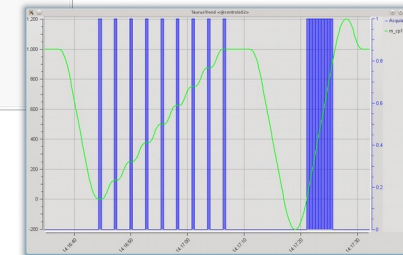
MacroServer – powerful sequencer

Device Pool – HW access + low level control

```

from sardana.macroserver.macro import macro

@macro()
def hello_world(self):
    """This is a hello world macro"""
    self.output("Hello, world!")
  
```





# Macros - Python function or classes

Hooks

```

Door_1 [8]: loop 0 10 3
Starting loop
At step 0
running hook with hints=['pre-acq']
En hook 1
At step 3
running hook with hints=['pre-acq']
En hook 1
At step 6
running hook with hints=['pre-acq']
En hook 1
At step 9
running hook with hints=['pre-acq']
En hook 1
Finished loop

```

Input parameters & results & data

SPEC like commands

```

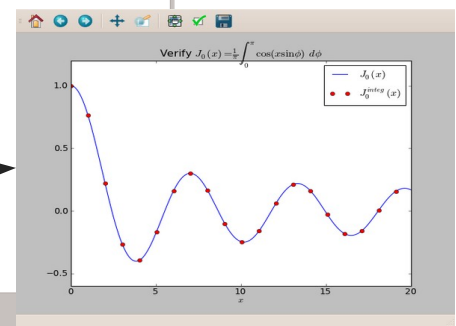
13
14
15 @macro([ ["moveable", Type.Moveable, None, "moveable to move"],
16          ["position", Type.Float, None, "absolute position"] ])
17 def move(self, moveable, position):
18     """This macro moves a motor to the specified position"""
19     moveable.move(position)
20     self.output("Motor ended at ", moveable.getPosition())
21
22 class Loop(Macro, Hookable):
23     """A macro that executes a for loop. It accepts hooks.
24     """
25     hints = {'allowsHooks': ['pre-move', 'post-move', 'pre-acq', 'post-acq']}
26     param_def = [['start', Type.Integer, None, 'start point'],
27                 ['stop', Type.Integer, None, 'end point'],
28                 ['step', Type.Integer, 1, 'step']]
29     result_def = [['result', Type.Integer, None, 'result']]
30
31     def hook1(self):
32         self.output("In hook 1")
33
34     def run(self, start, stop, step):
35         self.info("Starting loop")
36         self.hooks = [ (self.hook1, ["pre-acq"]) ]
37         for i in xrange(start, stop, step):
38             self.output("At step %d" % i)
39             self.flushOutput()
40             for hook, hints in self.hooks:
41                 hook()
42             self.info("Finished loop")
43             return i
44
45 class hooked_scan(Macro):
46     """An example on how to attach hooks to the various hook points of a scan.
47     """
48     param_def = [
49         ["motor", Type.Moveable, None, 'Motor to move'],
50         ["start_pos", Type.Float, None, 'Scan start position'],
51         ["end_pos", Type.Float, None, 'Scan final position'],
52         ["nr_interv", Type.Integer, None, 'Number of scan intervals'],
53         ["integ_time", Type.Float, None, 'Integration time']]
54
55     def hook1(self):
56         self.info("\thook1 execution")
57
58     def run(self, mot, start, end, nr, intt):
59         self.ascan, pars = self.createMacro("ascan", mot, start, end, nr, intt)
60         self.ascan.hooks = [(self.hook1, ["pre-acq"])]
61         self.runMacro(ascan)
62
63     @property
64     def data(self):
65         return self.ascan.data
66
67 @macro()
68 def ask_number_of_points(self):
69     """Asks user for the number of points"""
70     nb_points = self.input("How many points?", data_type=Type.Integer)
71
72 @macro()
73 def J0_plot(self):
74     """Sample J0 at linspace(0, 20, 200)"""
75     x = linspace(0, 20, 200)
76     y = j0(x)
77     x1 = x[::10]
78     y1 = map(j0i, x1)
79
80     self.pyplot.plot(x, y, label='J0(x)')
81     self.pyplot.plot(x1, y1, 'ro', label='J0-integ(x)')
82     self.pyplot.title('Verify J0(x)=\frac{1}{\pi} \int_0^x \cos(\sin \theta) d\theta')
83     self.pyplot.xlabel('$x$')
84     self.pyplot.legend()
85
86
87

```

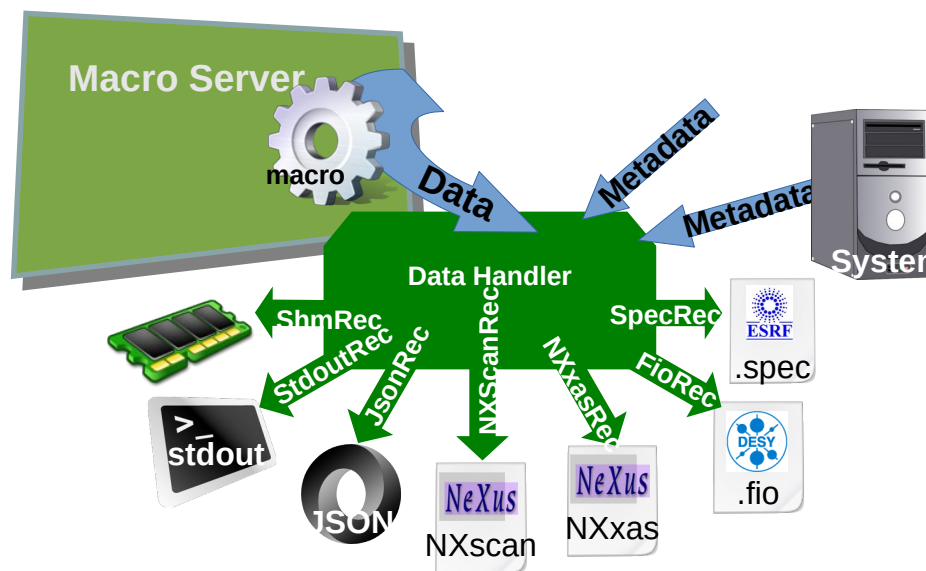
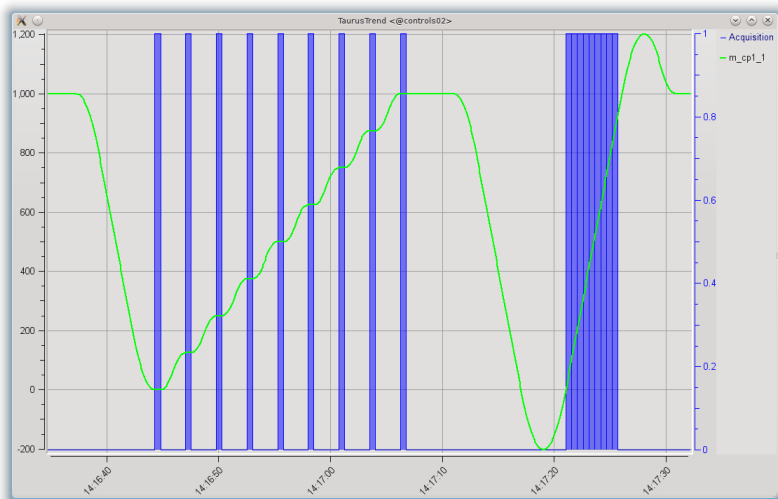
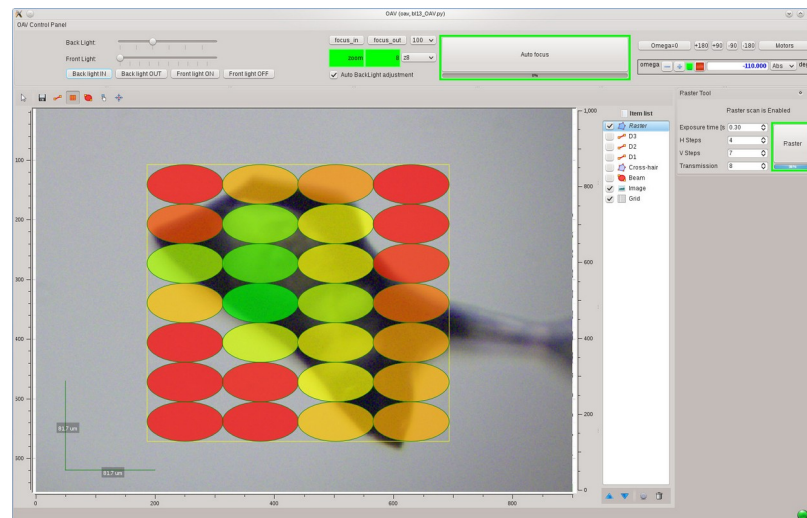
Adding, editing macros at runtime

Interactive macros

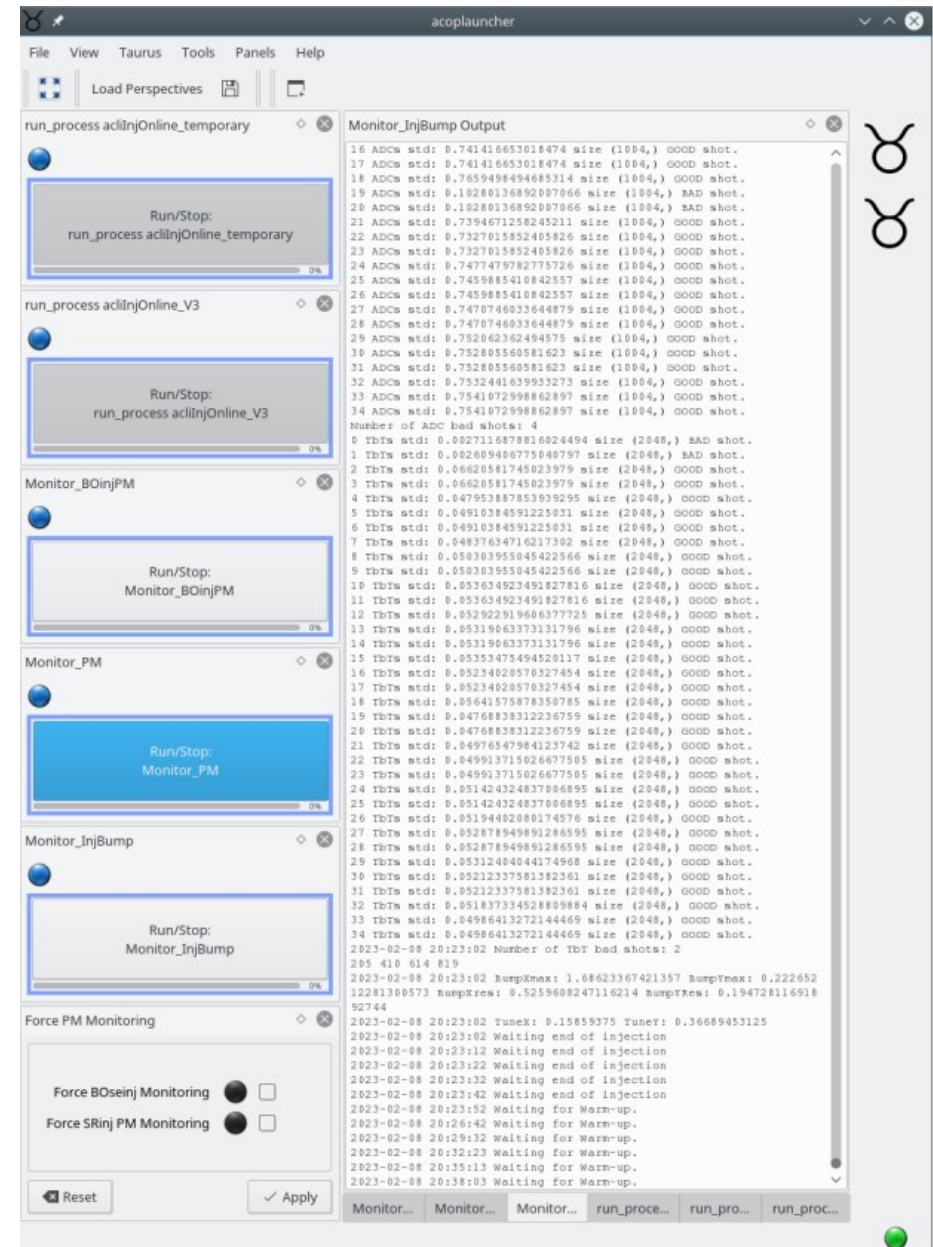
Plotting



- Macros for experiment control:
  - Scans: step & continuous
  - Optical elements alignments
  - Experiment sequences
  - Sample environment preparation



- Macros to support accelerators operation and monitor:
- LINAC injection parameters
- Injection to Booster
- Injection to SR
- Injection bump





# Installation

- PyPI
- Debian Linux
- **conda**

```
conda create -c conda-forge -y -n macros_tutorial \
  sardana \
  matplotlib \ # macro plotting
  h5py \       # HDF5 file recorder
  qtconsole \  # QtSpock
  jupyter_client=6.1.11 \ # pinned version for QtSpock
  pyqtgraph \ # scan plotting
  silx \       # opening scan file
  tango-test   # for demo purposes
conda activate macros_tutorial
pip install git+https://github.com/ALBA-Synchrotron/sardana-tango.git
```

More on: [https://sardana-controls.org/users/getting\\_started.html](https://sardana-controls.org/users/getting_started.html)



# Contents

- **Introduction**
  - What is Sardana and MacroServer
  - Usage of macros
  - Installation
- **Macro development**
  - Create simulated environment
  - Macro execution clients
  - Hello world example
  - Macro features
- **Scan with Tango attributes**





# Server, Spock and sar\_demo...

```
# Start the Sardana server
```

```
Sardana demo1
```

```
# Confirm with [Y]
```

```
# Start the Spock client - SPOC, based on IPython
```

```
Spock
```

```
# Confirm with [y]
```

```
# select the door Door_demo1_1
```

```
# In Spock create simulated elements:
```

```
sar_demo
```

```
# List created elements
```

```
lsa
```

```
# Execute a step scan
```

```
ascan mot01 0 5 5 0.1
```

```
Door_demo1_1 [7]: lsa
```

Name	Type	Controller	Axis
ct01	CTExpChannel	ctctrl01	1
ct02	CTExpChannel	ctctrl01	2
ct03	CTExpChannel	ctctrl01	3
ct04	CTExpChannel	ctctrl01	4
gap01	PseudoMotor	slitctrl01	1
ior01	IORegister	iorctrl01	1
ior02	IORegister	iorctrl01	2
ioveri001	PseudoCounter	ioveri0ctrl01	1
mot01	Motor	motctrl01	1
mot02	Motor	motctrl01	2
mot03	Motor	motctrl01	3
mot04	Motor	motctrl01	4
offset01	PseudoMotor	slitctrl01	2
oned01	OneDExpChannel	onedctrl01	1
tg01	TriggerGate	tgctrl01	1
twod01	TwoDExpChannel	twodctrl01	1
zerod01	ZeroDExpChannel	zerodctrl01	1
zerod02	ZeroDExpChannel	zerodctrl01	2
zerod23	ZeroDExpChannel	zerodctrl01	3
zerod24	ZeroDExpChannel	zerodctrl01	4



# How does it work?

- Check macro code with:  
**prdef sar\_demo**
- Macro is available in Spock as a magic command.
- Macros code is executed on the server side.

```
Door_demo1_1 [9]: prdef sar_demo
@macro([
  ["elements",[
    ["elem_type", Type.String, None, "Element type"],
    ["elem_quant", Type.Integer, 0, "Element quantity"],
    {'min': 0, 'max': None}
  ], None, "Number of elements to be created per type"
])
def sar_demo(self, elements):
  """Sets up a demo environment. It creates many elements for testing"""

  try:
    SAR_DEMO = self.getEnv(_ENV)
    self.error("A demo has already been prepared on this sardana")
    return
  except:
    pass

  db = PyTango.Database()

  elements_quant = default_elements_quant.copy()
  for elem_type, elem_quant in elements:
    elem_type_lower = elem_type.lower()
    if elem_type_lower not in elements_quant:
      raise ValueError(
        "element type '{}' is not recognised (allowed types: {})".format(
          elem_type, list(elements_quant.keys()))
      )
    # Replace the default quantity with the configured one
    elements_quant[elem_type_lower] = elem_quant

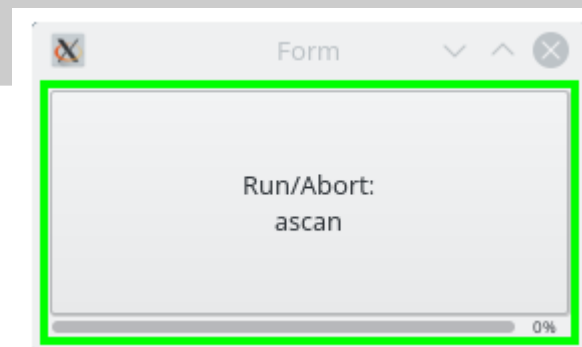
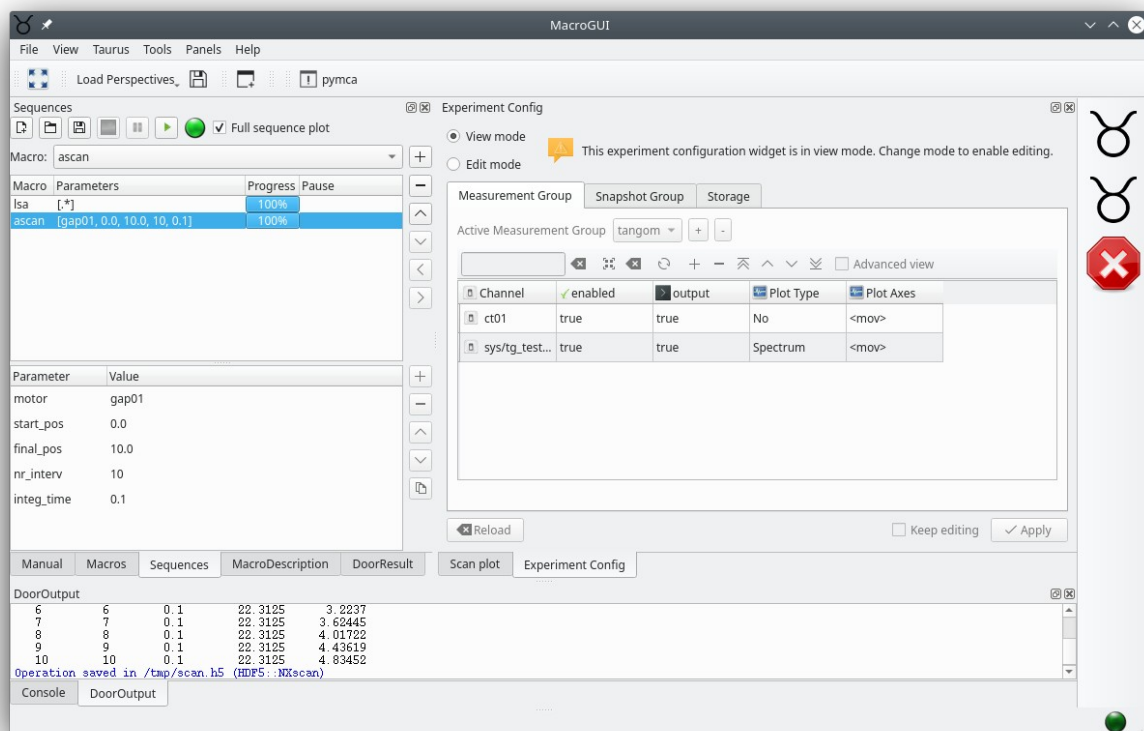
  mot_ctrl_name = get_free_names(db, "motctrl", 1)[0]
  ct_ctrl_name = get_free_names(db, "ctctrl", 1)[0]
```

# Start the Macro execution GUI

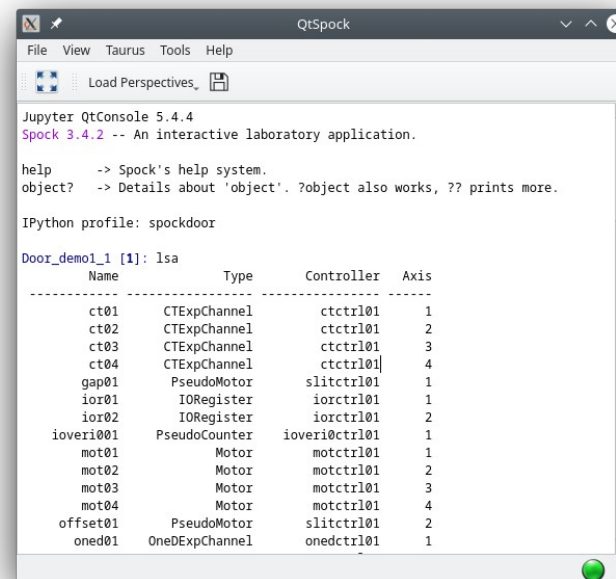
**taurus gui macrogui**

# Connect to the MacroServer and Door: Menu → Taurus → Macro execution configuration...

# Compose and run a sequence



Macro Button



QtSpock



# How to plug-in a macro?

- Macro discovery:
  - The plugin discovery system is based on directory scanning and python module inspection
  - Custom macros should be installed in one of the MacroPath directories

```
# Create a directory for you macros:
```

```
mkdir /tmp/macros
```

```
# In Spock set the MacroPath property:
```

```
_MACRO_SERVER.put_property({"MacroPath":["/home/local/zreszela/  
miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/  
sardana/macroserver/macros/examples/", "/tmp/macros"]})
```

```
# Restart the Sardana server and Spock
```

```
# List newly added macro libraries
```

```
lsmacLib
```

Name	Location
communication	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/communication.py
demo	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/demo.py
discrete	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/discrete.py
env	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/env.py
acquisition	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/acquisition.py
debug	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/debug.py
funcs	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/funcs.py
hooks	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/hooks.py
motion	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/motion.py
parameters	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/parameters.py
plotting	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/plotting.py
scans	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/scans.py
specific_experiments	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/specific_experiments.py
submacros	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/submacros.py
user_input	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/examples/user_input.py
expconf	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/expconf.py
expert	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/expert.py
h5storage	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/h5storage.py
hkl	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/hkl.py
lorelist	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/lorelist.py
lists	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/lists.py
mca	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/mca.py
scan	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/scan.py
sequence	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/sequence.py
standard	/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.10/site-packages/sardana/macroserver/macros/standard.py



# Hello world example

- Macro functions are decorated with `@macro()`
- Macros log messages are sent to the clients
- Different log levels can be used
- Macros may use input parameters and results
- >50 parameter types are available

```
# In Spock edit the macro:
```

```
edmac hello salute
```

```
# Apply the new code with [y]
```

```
# Run the macro:
```

```
hello
```

```
# Let's add a macro parameter of type String called "name"
```

```
edmac hello
```

```
Door_demo1_1 [13]: prdef hello
@macro(["name", Type.String, "Cape Town", "Whom to greet?"])
def hello(self, name):
    """"Macro hello""""
    self.output(f"Hello {name}!")
```



# Macro features

- Macros can **call other macros**: `self.<macro_name>()` e.g. `self.hello("Paul")`
- **Interactive macros** - macros can ask for input e.g. `ask_peak`.
- **Plotting** – macro can plot data e.g. `mandelbrot 10 100`
- **Interrupting** – macros can be stopped, aborted, paused and resumed. Run a scan macro and try `<Ctrl+C>`.
- Macros can access to a shared **environment**: `lsenv`, `senv`, `usenv`

```
# In Spock define MultiGreet environment variable
senv MultiGreet 3
# Edit the macro and repeat greeting MultiGreet times
edmac hello
# Apply the new code with [y]
# Run the macro:
hello
```

```
Door_demo1_1 [36]: prdef hello
@macro([[ "name", Type.String, "Cape Town", "Whom to greet?" ]])
def hello(self, name):
    """Macro hello"""
    for _ in range(self.getEnv("MyEnvVar")):
        self.output(f"Hello {name}!")
```

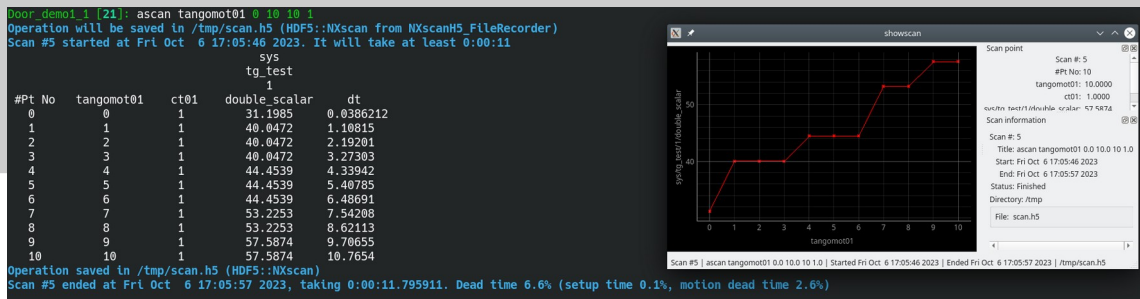


# Contents

- **Introduction**
  - What is Sardana and MacroServer
  - Usage of macros
  - Installation
- **Macro development**
  - Create simulated environment
  - Macro execution clients
  - Hello world example
  - Macro features
- **Scan with Tango attributes**

# Scan with Tango attributes

```
# In Spock set the PoolPath property and restart the server
Pool_demo1_1.put_property({"PoolPath":
"/home/local/zreszela/miniconda3/envs/macros_tutorial/lib/python3.1
0/site-packages/sardana_tango/ctrl/"})
# Define the controller instance
defctrl TangoAttrMotorController tangomotctrl01
# Define the motor instance
defelem tangomot01 tangomotctrl01 1
# Configure the motor to point to a Tango attribute
tangomot01.TangoAttribute = "sys/tg_test/1/ampli"
# In expconf define a measurement group with ct01
sys/tg_test/1/double_scalar, configure plotting and storage
expconf
# Open online plotting widget
showscan
# Run a scan
ascan tangomot01 0 10 10 1
# Open the data file
silx view /tmp/scan.h5
```







# Home Page

<http://www.sardana-controls.org>

## Access to:

- Documentation
- Releases
- Git repository
- Mailing lists
- Bugs & Requests tracker
- Enhancement Proposals
- ...

