Closed-loop electrophysiological experiments and automated metadata acquisition with RELACS

Jan Benda & Jan Grewe
LMU München, Department Biologie II, Großhadernerstr. 2, 82152 Planegg-Martinsried, Germany

33th Göttingen Neurobiology Conference, 2011

1 Introduction

relacs
... enjoy your recordings

RELACS ("**R**elaxed **EL**ectrophysiological data **A**cquisition, Control, and **S**timulation") is a fully customizable software platform for data acquisition, online analysis, and stimulus generation specifically designed for electrophysiological recordings. The main design goals are (i) **closed loop** recordings on both the stimulus time-scale ($> 10\,\text{ms}$) and the sampling time-scale ($< 1\,\text{ms}$), i.e. **dynamic clamp**, (ii) a hardware independent **modular** design, and (iii) automatic annotation of the data with **metadata**.

2 Free Software

RELACS is free and open software published under the GPL to foster development and exchange of innovative experimental protocols and analysis techniques.

RELACS is independent of any specific hardware manufacturer.

RELACS is programmed in C++ (more then 160 000 lines of code), using multithreaded Qt for the GUI, and runs on Linux.

Data recorded with RELACS are published in 15 scientific papers in journals like Neuron, Nature, Neuroscience, Journal of Neurosciene, Journal of Neurophysiology, etc.

For more information and downloads visit

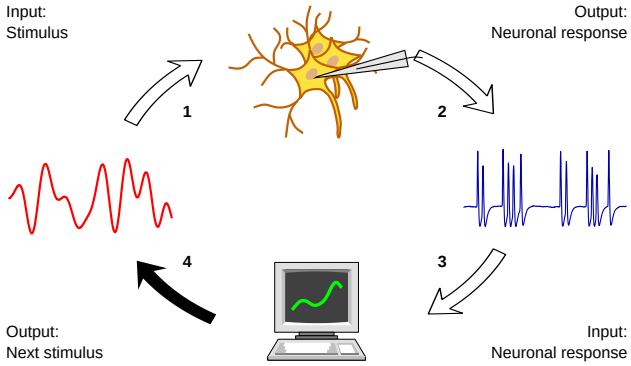
www.relacs.net

3 Closed-Loop Experiments

In a closed-loop experiment

1. a stimulus is presented,
2. the resulting response
3. is analyzed immediately, and
4. properties of the next stimulus (mean intensity, standard deviation, spectral content...) are adjusted as needed.

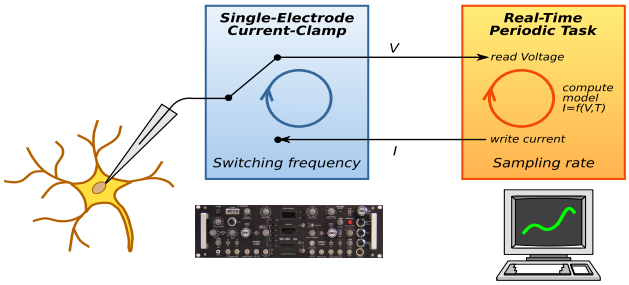
RELACS is designed as a framework for closed-loop experiments.



See our review "From response to stimulus: adaptive sampling in sensory physiology" for more details (Benda et al. 2007, Current Opinion in Neurobiology 17(4): 430-436).

4 Dynamic Clamp

The dynamic clamp is a closed-loop experiment on a per sample basis (tens of kHz) where each sampled value of the cell's membrane potential is used to compute a current that is injected back into the cell.



RELACS supports software dynamic clamp, i.e. no additional hardware is needed, that is implemented as an RTAI real time Linux kernel module. (in collaboration with H. R. Polder, npi electronic GmbH, Tamm, Germany)

5 Modular Design and Hardware Independence

The central building blocks of RELACS are the “Research Protocols”. These are freely programmable C++ plugins that

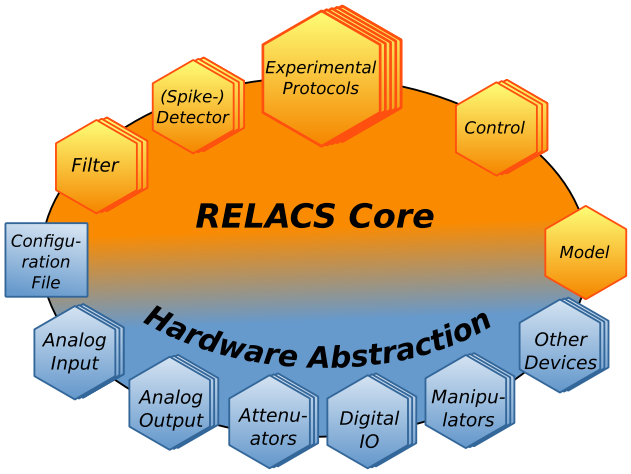
- take recorded and preanalyzed data
- perform analysis & display results
- generate next stimulus

Hardware of different manufacturers is integrated by

- device plugins and
- a configuration file

into RELACS, providing a hardware independent interface for

- filter and detectors,
- control modules, and
- research protocols.



Therefore, research protocols

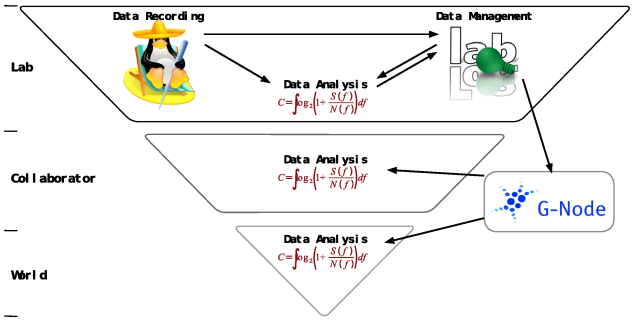
- can be used on all the different experimental setups in your lab without any modifications
- can be shared with other labs

5.1 Code example

A minimal implementation of a research protocol looks like this:

```
int Example::main( void ) {
  double frequency = number( "frequency" );
  double duration = number( "duration", "s" );
  double amplitude = 0.0;
  OutData signal;
  signal.setTrace( "LeftSpeaker" );
  signal.sineWave( frequency, duration, amplitude );
  SampleDataD rate( 0.0, duration, 0.001 );
  for ( int counter=0; counter<20; counter++ ) {
    write( signal );
    sleep( duration + pause );
    EventData spikes( events( "Spikes-1" ),
                      signalTime(), signalTime() + duration );
    double meanrate = spikes.rate( 0.3*duration, duration );
    spikes.addRate( rate, counter, GaussKernel( sigma ) );
    P.lock();
    P.clear();
    P.setXRange( 0.0, duration );
    P.plot( rate, 1000.0, Plot::Yellow, 2, Plot::Solid );
    P.draw();
    P.unlock();
    if ( meanrate < targetrate ) {
      amplitude *= 2.0;
      signal.sineWave( frequency, duration, amplitude );
    }
  }
  return Completed;
}
```

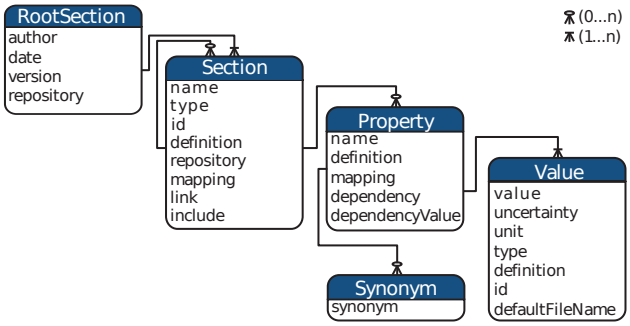
6 Exchange metadata



- All data transfer for analysis, mamagement, and sharing requires talking about data.
- How to exchange metadata?
- How to record metadata?

6.1 *od*ML — open metadata markup language

- simple key-value based, hierarchical structure:



- all meta-data can be immediately stored (e.g. no XML namespace extensions required)
- independent of data-base schemas
- standardization through terminologies

7 *od*ML Terminologies

The flexibility of *od*ML allows any keywords and sections to be used.
However, to ensure interoperability some standard is needed. This is achieved by the *od*ML terminologies, that provide standard names (keys) and sections with definitions.
The terminologies are not ment to be fixed, but rather should grow based on input from the user community.

7.1 How to use *od*ML?

1. Assemble properties:
   - If you find an appropriate property in the *od*ML terminologies, use it!
   - Ignore all properties that do not match.

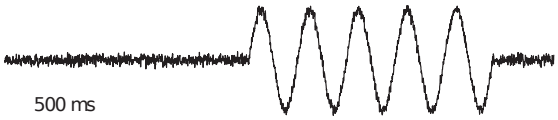
- Add your own properties that are not yet in the terminology, if possible with a description.

2. Write them into an *od*ML XML file
3. Transfer them to an analysis or database program

⇒ *od*ML flexibility: all available metadata can be immediately stored in a file

⇒ *od*ML standard: The database of the German-Neuroinformatics-Node is based on *od*ML: www.g-node.org/odml (see poster #T27-10B)

7.2 Example: Stimulus Description




```
odML
└── MyStimulus –[stimulus]
        –Duration = 2.25 s
        –Repetitions = 25
        –InterstimulusInterval = 5 s
        –Modality = visual
        –OutputChannel = LED1
    ├── DC –[stimulus/dc]
    │       –TemporalOffset = 0.0 s
    │       –Intensity = 10.000 photons/s
    │       –Duration = 2.25 s
    ├── WhiteNoise –[stimulus/white_noise]
    │       –Duration = 2.25 s
    │       –TemporalOffset = 0.0 s
    │       –MeanIntensity = 0.0 photons/s
    │       –StandardDeviation = 350 photons/s
    │       –UpperCutoffFrequency = 256 Hz
    └── Sinewave –[stimulus/sine_wave]
            –Duration = 1.0 s
            –TemporalOffset = 1.0 s
            –MeanIntensity = 0.0 photons/s
            –Amplitude = 5.000 photons/s
            –Frequency= 5 Hz
            –Phase = 0
```

8 Automated Recording of Metadata

- Every online recording software knows about most of the important meta-data!

⇒ All available Meta-data should be written to a file directly from the recording software, if possible using *od*ML terminologies.

- Such automated meta-data storage is the basis for making public data bases, such as www.g-node.org, work.

We provide libraries for reading and writing *od*ML files for Java, MatLab, and C++. A Python library is on its way.

RELACS records many meta-data:

- General infos about the experiment (from the dialog)
- Main characteristics of the recorded cell
- All RELACS-controlled hardware settings (e.g. sampling rate)
- All settings and version numbers of the research protocols
- Properties of the stimuli

Automated transfer to the LabLog database (see Poster #T27-10A)

and/or the G-node database (see poster #T27-10B) is thus possible.