RELACS — a modular software platform for closed-loop experiments

Jan Benda

Department Biologie II
Ludwig-Maximilian
Universität München

Sensory electrophysiology

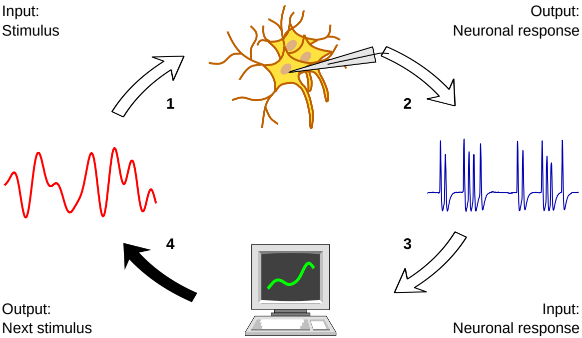
Electrosensory systems of weakly eletric fish

Auditory system of grasshopper and crickets

How are sensory stimuli processed by sensory systems?

Closed-loop experiments with RELACS

1. Present a stimulus
2. Record the response
3. Immediately analyze and visualize the data
4. Generate the next stimulus



Simple closed-loop experiments

- Online visualization of processed data:
  - General infos, e.g. quality of spike detection, sensitivity of the cell, temperature, condition of animal, ...
  - Specific results, e.g. spike raster, firing rates, spike-triggered averages, ...

⇒ Speeds up manual (“traditional”) closed-loop

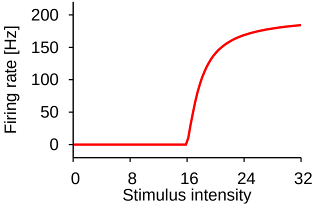
Simple closed-loop experiments

- Online visualization of processed data:
  - General infos, e.g. quality of spike detection, sensitivity of the cell, temperature, condition of animal, ...
  - Specific results, e.g. spike raster, firing rates, spike-triggered averages, ...

  $\Rightarrow$ Speeds up manual (“traditional”) closed-loop
- Set stimuli relative to the neuron’s dynamic range
- Automatically control motorized electrodes (great for dual unit recordings!)
- Optimize tuning curve measurements
- ...

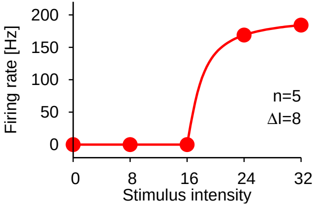
Example: tuning curve measurement

Traditional:



Example: tuning curve measurement

Traditional:

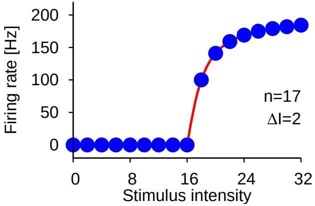


either:

fast ⟶ low resolution

Example: tuning curve measurement

Traditional:



either:

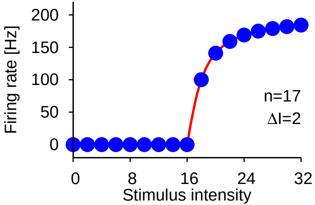
fast ⟶ low resolution

or:

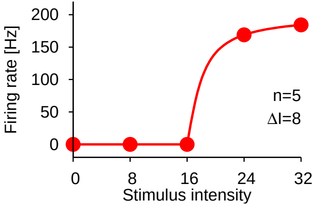
high resolution ⟶ slow

Example: tuning curve measurement

Traditional:



Closed loop:



either:

fast ⟶ low resolution

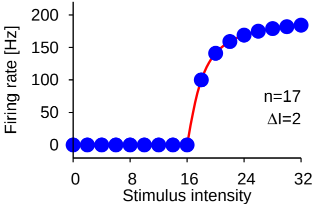
or:

high resolution ⟶ slow

1. start with low resolution

Example: tuning curve measurement

Traditional:



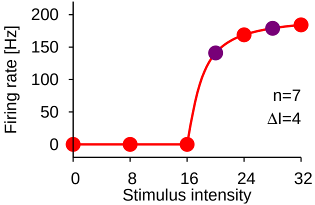
either:

fast ⟶ low resolution

or:

high resolution ⟶ slow

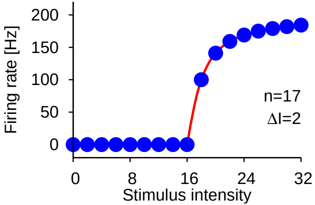
Closed loop:



1. start with low resolution
2. increase resolution where necessary!

Example: tuning curve measurement

Traditional:



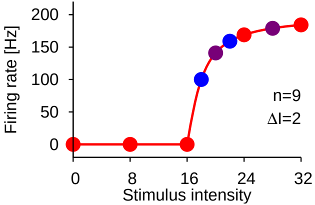
either:

fast ⟶ low resolution

or:

high resolution ⟶ slow

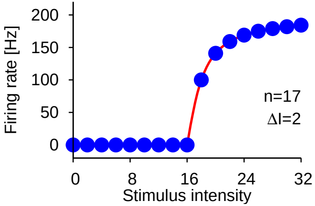
Closed loop:



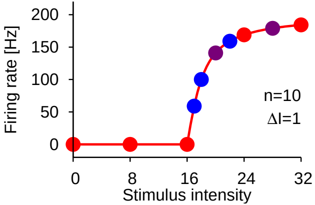
1. start with low resolution
2. increase resolution where necessary!
3. further increase resolution

Example: tuning curve measurement

Traditional:



Closed loop:



either:

fast $\longrightarrow$ low resolution

or:

high resolution $\longrightarrow$ slow

1. start with low resolution
2. increase resolution where necessary!
3. further increase resolution

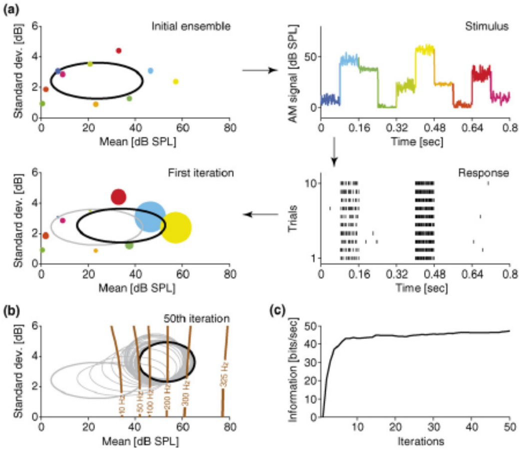
Advanced closed-loop experiments

New experimental designs are possible:

- Optimal search for a neuron's receptive field.
- Search for stimuli that drive a neuron in an "optimal" way.
- Find set's of stimulus parameter that result in the same response (iso-response method).
- ...

Benda et al. (2007): "From response to stimulus: adaptive sampling in sensory physiology." *Curr. Opin. Neurobiol.* **17**: 430–436.

Example: optimal stimulus ensembles

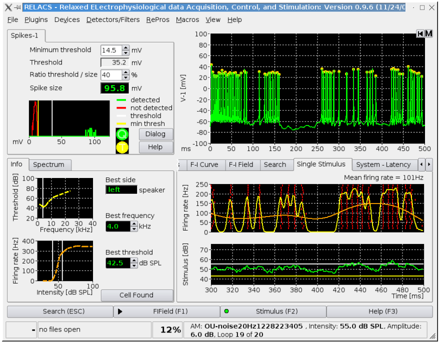


Machens et al. (2005) *Neuron* **17**: 47–56.

RELACS ... enjoy your recordings

Relaxed Electrophysiological data Acquisition, Control, and Stimulation

RELACS is a framework for closed-loop experiments



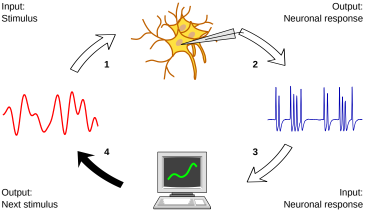
⇒ currently 15 scientific publications based on RELACS data in *Neuron, J Neurosci, PLoS Biol, Nat Neurosci, J Neurophysiol, etc.*

RELACS research protocols

In RELACS the closed-loop cycle can be freely programmed as a C++ plugin (“research protocol”).

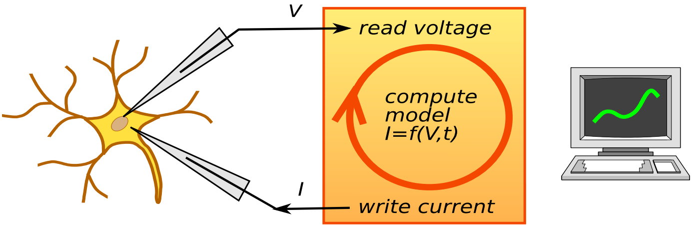
The research-protocol plugins

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



Dynamic clamp

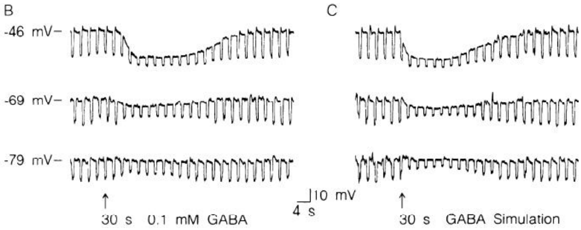
Current-clamp, with the current I computed as a function of the measured membrane potential V.



Closed-loop at a per sample time scale (tens of kHz).

Artificial conductances

$$I = g(t) \cdot (V - E)$$

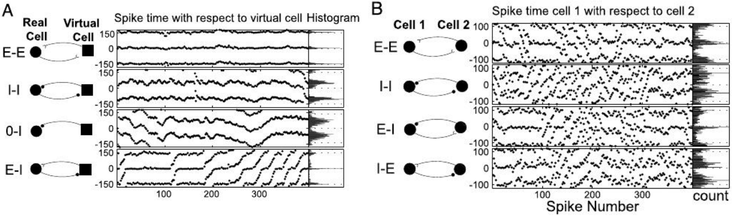


Andrew A. Sharp, Michael B. ONeil, L. F. Abbott, & Eve Marder (1993) *J Neurophysiol*

- Synaptic conductances
- Voltage-gated conductances

Artificial networks

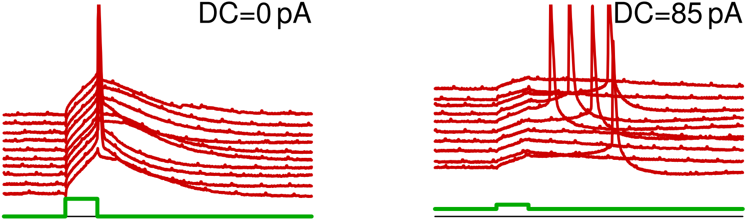
$$I_1 = g_{syn}(V_2)\cdot(V_1 - E) \qquad I_2 = g_{syn}(V_1)\cdot(V_2 - E)$$



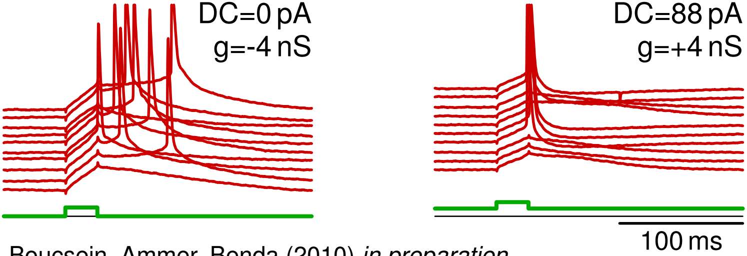
Theoden I. Netoff, Matthew I. Banks, Alan D. Dorval, Corey D. Acker, Julie S. Haas, Nancy Kopell, & John A. White (2005) *J Neurophysiol*

- Artificially couple real neurons
- Couple with simulated neurons

Precision-switch by leak conductance



Dynamic clamp: leak current $I = g(V - E)$, $E := V_{rest}$



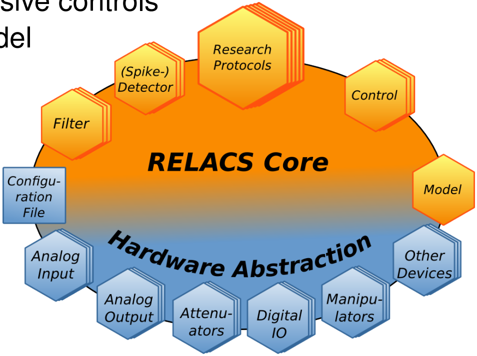
Boucsein, Ammer, Benda (2010) *in preparation*

relacs

Modular design

RELACS core with flexible C++ Plugins for

- hardware abstraction
- data pre-processsing (filter, spike detectors)
- **research protocols**
- passive controls
- model

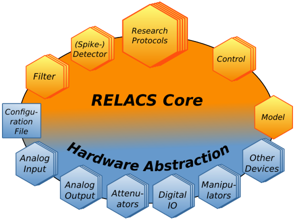


Hardware independent protocols

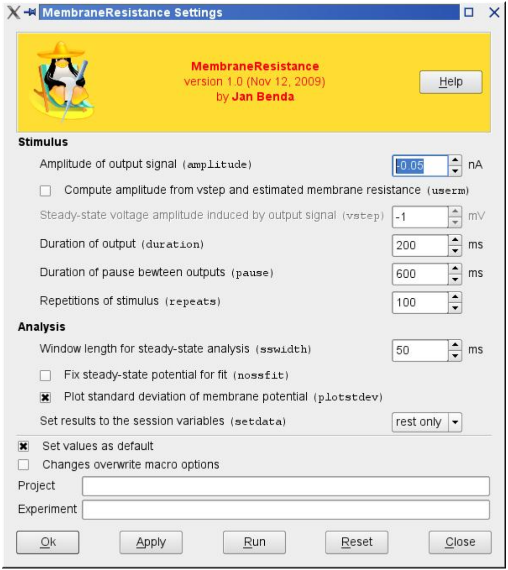
RELACS integrates all hardware components.

Research protocols for RELACS

- are implemented independently of specific hardware
- can be used on all the different experimental setups in your lab without any modifications
- can be shared with other labs



Options for research protocols



Macros

... execute **research protocols** with specific parameter settings:

```
$FIField startsession
FIField
SysLatency
FICurve: duration=40ms; pause=260ms;
detector Spikes-1: save
```

Research-protocol example

```
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;
}
```

C++ library for data analysis

Data structures (classes, container):

- *Array* — Basic 1-D vector
- *SampleData* — 1-D data vector with regularly sampled time axis
- *Map* — Sequence of $x|y$ data pairs

Algorithms:

- basic statistics (moments, quartiles, histogram)
- power spectra, coherence, transfer function
- linear fits
- non-linear fits (Simplex, Levenberg-Marquardt)

C++ library for data analysis

Data structures (classes, container):

- *EventData* — Spikes and other point process data
- *EventList* — Multi-trial spike trains

Algorithms:

- firing rates (mean, PSTH binned/kernel, 1/ISI)
- CV, Fano factor, ISI correlation
- vector strength, reliability, jitter
- mutual information (lower and upper bound)

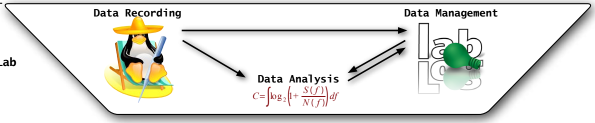
Simulation mode

Research protocols also run on simulated data:

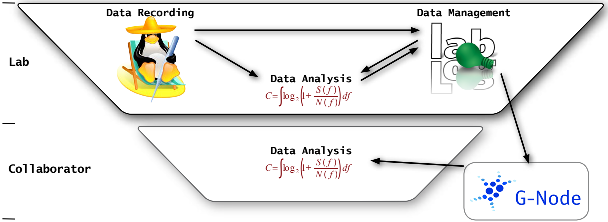
- test closed-loop algorithms
- directly compare models with experimental data

Meta-data: the data-chain

Lab



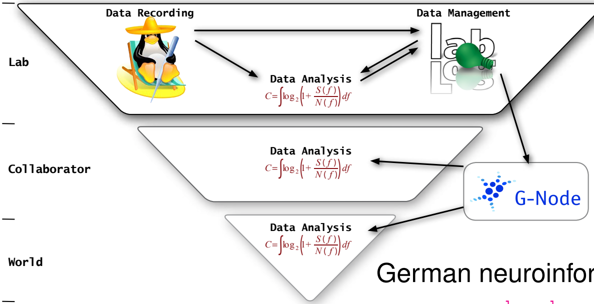
Meta-data: the data-chain



German neuroinformatics node

www.g-node.de

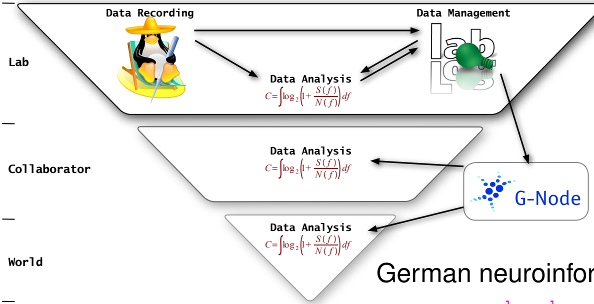
Meta-data: the data-chain



German neuroinformatics node
www.g-node.de

- All data transfer for analysis, mamagement, and sharing requires talking about data.

Meta-data: the data-chain



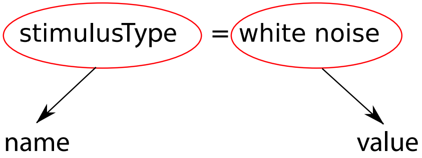
German neuroinformatics node
www.g-node.de

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

The meta-data problem

Name-value (+unit) pairs for:

- Stimuli
- Experimental settings
- Cell, preparation, experimental subject
- Hardware properties
- Analysis parameter
- etc.



The meta-data problem

Name-value (+unit) pairs for:

- Stimuli
- Experimental settings
- Cell, preparation, experimental subject
- Hardware properties
- Analysis parameter
- etc.

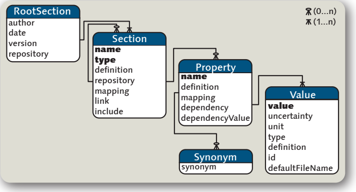
But:

- What name to choose?
- What does it mean?
- How to share meta-data?

odML — a proposal

— 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

Grewe, Wachtler, Benda (2010) *submitted*. www.g-node.org/odml

odML terminologies

names & definitions

Hardware:

Amplifier:

name	type	description
Gain	float	The amplifier gain.
HighpassCutoff	float	The cutoff frequency of the amplifier's highpass filter. Given in Hz.
LowpassCutoff	float	The cutoff frequency of the amplifier's lowpass filter. Given in Hz.
Mode	string	The amplifier mode. E.g. Bridge, CC, VC etc.

How to use odML?

1. Assemble properties:
   - If you find an appropriate property in the odML-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.

How to use odML?

1. Assemble properties:
   - If you find an appropriate property in the odML-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 odML XML file

How to use odML?

1. Assemble properties:
   - If you find an appropriate property in the odML-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 odML XML file
3. Transfer them to an analysis or database program

How to use odML?

1. Assemble properties:
   - If you find an appropriate property in the odML-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 odML XML file
3. Transfer them to an analysis or database program

$\Rightarrow$ odML flexibility: all available metadata can be immediately stored in a file

How to use odML?

1. Assemble properties:
   - If you find an appropriate property in the odML-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 odML XML file
3. Transfer them to an analysis or database program

⇒ odML flexibility: all available metadata can be immediately stored in a file

⇒ odML standard: The G-Node electrophysiology database is based on odML: `ww.g-node.org`

How to record meta-data?

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

How to record meta-data?

- 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 odML terminologies.

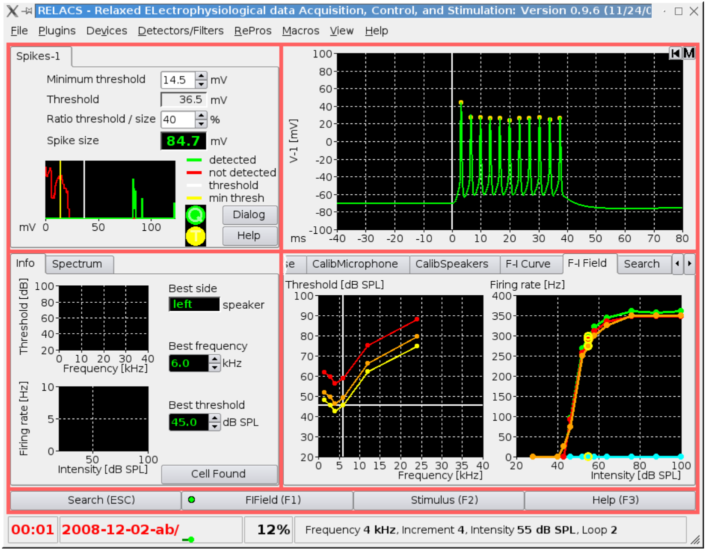
How to record meta-data?

- 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 odML terminologies.

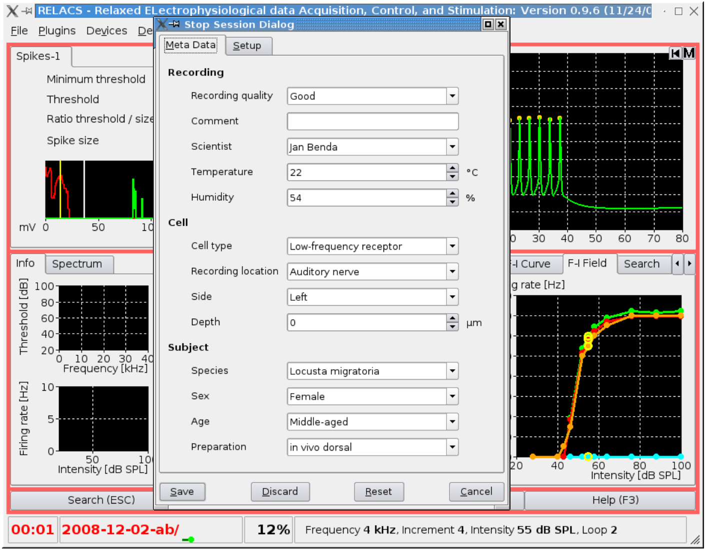
- Such automated meta-data storage is the basis for making public data bases, such as `www.g-node.org`, work.

Meta-data acquisition by RELACS



relacs

Meta-data acquisition by RELACS



relacs

Meta-data acquisition by RELACS

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

relacs

... enjoy your recordings

→ Closed-loop experiments

→ Dynamic clamp

→ Simulation mode

→ Hardware independent

→ Data analysis libraries

→ Meta-data storage

→ Open source, GPL, Linux

→ ~ 160 000 lines of C++ code

by Jan Benda

www.relacs.net