Lab::Measurement – measurement control with Perl

S. Reinhardt[1], C. E. Lane[2], C. Butschkow[1], A. Iankilevitch[1], A. Dirnaichner[1], and A. K. Hüttel[1]

[1] *Institute for Experimental and Applied Physics, University of Regensburg, 93040 Regensburg, Germany*
[2] *Department of Physics, Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104, USA*

Flexible measurement needed?!

- Tired of following your wires in square meters of LabVIEW diagrams?
- Tired of clumsy string handling and low-level driver functions in your looong C program?
- Use a text processing language to manage your measurement! Use Perl!

```
# Read out SR830 Lock In Amplifier at GPIB address 13
use warnings;
use strict;
use 5.010;
use Lab::Measurement;

my $lia = Instrument(
    'SR830',
    {
        connection_type => 'LinuxGPIB',
        gpib_address    => 13,
    }
);

my $amp = $lia->get_amplitude();
say "Reference output amplitude: $amp V";

my $freq = $lia->get_frequency();
say "Reference frequency: $freq Hz";

my ( $r, $phi ) = $lia->get_rphi();
say "Signal: r=$r V   phi=$phi degree";
```

Currently supported hardware

Hardware driver backends:

- NI-VISA (both on MS Windows and on Linux) and all hardware supported by it
- LinuxGPIB and all hardware supported by it
- Linux USB-TMC kernel driver
- Oxford Instruments IsoBus
- TCP connection, generic network socket
- Serial port, USB serial connection

Growing number of high-level drivers (more are *very* easy to add):

- Multimeters: HP / Agilent / Keysight
- DC sources: Yokogawa / Keithley
- Lock-in amplifiers: Stanford Research / Signal Recovery
- Temperature controllers: Lakeshore / Oxford Instruments
- RF / microwave sources, spectrum analyzers, VNAs from Rohde & Schwarz
- and many more...

Key facts

- Open source / free software
- http://www.labmeasurement.de/
- License: same as Perl (GPL-1+ or Artistic)
- Releases on CPAN, development on Github
- Contributors and cooperations welcome!

Real world measurement

- Nested sweeps: gate voltage V_g, bias voltage V_{sd}
- For each point, read out multimeter, write values to data file
- Regularly updated “live” color plot

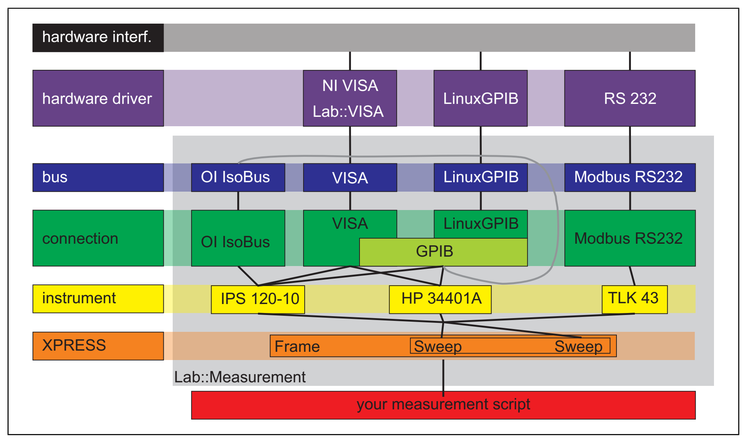
```
#-------- 1. Import Lab::Measurement -------

use warnings;
use strict;
use 5.010;

use Lab::Measurement;

#-------- 2. Some constants ----------------

my $gain          = -1e-9;     # A/V amplifier sensit.
my $stepwidthbias = 0.05;      # step width bias
my $stepwidthgate = 0.1;       # step width gate
my $NPLC          = 2;         # integration time, 2*20ms

#-------- 3. Initialize Instruments --------

my $gate_source = Instrument(
    'YokogawaGS200',
    {
        connection_type => 'VISA_GPIB',
        gpib_address    => 2,
        gate_protect    => 0
    }
);

my $bias_source = Instrument(
    'YokogawaGS200',
    {
        connection_type => 'VISA_GPIB',
        gpib_address    => 6,
        gate_protect    => 0
    }
);

my $DMM_I = Instrument(
    'HP34401A',
    {
        connection_type => 'VISA_GPIB',
        gpib_address    => 14,
        nplc            => $NPLC
    }
);

#-------- 4. Define the Sweeps -------------

my $bias_sweep = Sweep(
    'Voltage',
    {
        mode       => 'step',              # other options: list, continuous
        instrument => $bias_source,
        points     => [ -1, 1 ],           # starting point, end point
        stepwidth  => [$stepwidthbias],
        rate       => [1],                 # sweep rate to approach start point
        jump       => 1,                   # jump to next point, no sweep
        delay_before_loop => 3             # settle 3s before starting
    }
);

my $gate_sweep = Sweep(
    'Voltage',
    {
        mode       => 'step',
        instrument => $gate_source,
        points     => [ -0.5, 0.5 ],       # starting point, end point
        stepwidth  => [$stepwidthgate],
        rate       => [0.03],              # sweep rate to approach start point
        jump       => 1                    # jump to next point, no sweep
    }
);

#-------- 5. Create a DataFile -------------

my $DataFile = DataFile('DCDiamond');

$DataFile->add_column('Gate');
$DataFile->add_column('Bias');
$DataFile->add_column('Current');

$DataFile->add_plot(
    {
        'type'    => 'pm3d',
        'x-axis'  => 'Gate',
        'y-axis'  => 'Bias',
        'cb-axis' => 'Current',
        'refresh' => 'block'
    }
);

#-------- 6. Measurement Instructions -------

# Define the callback. This subroutine will be called at each point of the
# parameter grid.
my $my_measurement = sub {

    # this is run for each measurement point

    my $sweep = shift;

    # read_mode => 'cache' means use the value last written by Perl to the
    # device, but do not query the instrument.
    my $value1 = $gate_source->get_value( { read_mode => 'cache' } );
    my $value2 = $bias_source->get_value( { read_mode => 'cache' } );

    my $value3 = $DMM_I->get_value() * $gain;

    $sweep->LOG(
        {
            Gate    => $value1,
            Bias    => $value2,
            Current => $value3,
        }
    );
};

#-------- 7. Put everything together -------

$DataFile->add_measurement($my_measurement);

$bias_sweep->add_DataFile($DataFile);

my $frame = Frame();
$frame->add_master($gate_sweep);
$frame->add_slave($bias_sweep);

#-------- 8. And GO! -----------------------

$frame->start();
```

Output files

```
simon@keanae /tmp$ ls MEAS_000/
Config.txt  DCDiamond_01.png  DCDiamond.dat dc_diamond.pl
```

- Archival copy of the measurement script
- Measured data, tab-separated gnuplot format
- Live plot at end of measurement as png image

Internal architecture



- Modular structure. Easy to extend with new instrument drivers and connection types.
- Abstract IO layer, makes instrument drivers independent of hardware backends.

Recent improvements

- New drivers: R&S FSV spectrum analyzers, R&S ZVA and ZVM vector network analyzers, HP33120 function generator, Tektronix TDS2024 oscilloscope
- Automated testing of instrument drivers and high-level functionality via mock connection objects; continuous integration testing on both Linux (Travis CI) and Windows (Appveyor)
- New framework for drivers and connections based on **Moose** — de-facto standard for Perl5, enabling modern object oriented programming
  - Less boilerplate code, create classes in a descriptive way
  - Use **roles** for safe and efficient sharing of functionality between drivers
  - Organize SCPI subsystems into roles, which can be used by multiple drivers — see e.g. the new R&S FSV spectrum analyzer driver:

```
extends 'Lab::Moose::Instrument';

with qw(
    Lab::Moose::Instrument::Common

    Lab::Moose::Instrument::SCPI::Format

    Lab::Moose::Instrument::SCPI::Sense::Bandwidth
    Lab::Moose::Instrument::SCPI::Sense::Frequency
    Lab::Moose::Instrument::SCPI::Sense::Sweep

    Lab::Moose::Instrument::SCPI::Initiate

    Lab::Moose::Instrument::SCPIBlock

);
```

  - Extensive type system; easy validation of user provided function parameters.
- Access to all of gnuplot’s plot and curve options via PDL::Graphics::Gnuplot — both for live plots (qt, x11, wxt) and hardcopies (png, jpeg, svg, tikz, pdfcairo, ...)

Outlook

- Implement more drivers, connections, and XPRESS functionality with Moose
- Build portable, lightweight USB-TMC connection based on libusb

We gratefully acknowlegde funding by the DFG via the Emmy Noether grant Hu1808/1-1 “Carbon nanotubes as electronical and nano-electromechanical hybrid systems in the quantum limit”, the collaborative research centre SFB 689 “Spin phenomena in reduced dimensions”, and the graduate research school GRK 1570 “Electronic properties of carbon based nanostructures”.