Lab::VISA

& friends

VISA, GPIB, etc.

- Instruments can be connected in various ways:
  Serial port, GPIB, VXI, TCP/IP, etc.

- **GPIB** (hardware and software)
  - GPIB (IEEE488): Standard by Hewlett-Packard
  - Physical layer IEEE488.1
  - Command layer IEEE488.2
  - SCPI (Standard Commands for Programmable Instruments)

- **VISA** (software)
  - Virtual Instrument Software Architecture
  - VXI, GPIB, serial, or computer-based instruments
  - NI-VISA library is one implementation of the VISA standard

Lab::VISA

- Lab::VISA is software to control instruments in the lab via VISA (e.g. voltage sources, multimeters, etc.)

- Alternative to LabView, Gpplus, etc.

- But very different: no GUI, just API

Lab::VISA design goals

- **Flexible**
  - Allow any kind of measurement procedure
  - Control anything that has GPIB or serial connection
- **Safe**
  - Make sure you never drive voltage sources to fast and destroy your gates
- **Helpful**
  - Automatically collect additional information about the data (“metadata”)
  - User should have to type in additional information only once and then never again

Lab::VISA is in Perl

- Lab::VISA is written in Perl and C
- To be used from programs written in Perl

- Perl: interpreted scripting language
- Runs on almost every OS
- Extremely good in reading data files, manipulating data, etc.
- Allows to write quick and dirty scripts that get the job done
- Also allows to write clean fullsize programs

=> ideal for experimental physics

Lab::VISA architecture

- Divided into three parts
- Build on top of each other
- Provide increasing comfort
- Measurement scripts can be based on any of these stages

- **Lab::VISA** is the lowest layer. It makes the NI-VISA library accessible from perl and therefor allows to make any standard VISA call

- **Lab::Instrument** package makes communication with instruments easier by silently handling the protocol involved

- **Lab::Tools** is the highest abstraction layer. Provides support for writing good measurement scripts. Offers means of saving data and related meta information to disk, plotting data, etc.

Lab::VISA architecture

your (tiny) measurement program / script

```
+-------------------------+ +---------------+ +------------------+
|Lab::Instrument::HP34401A| |L::I::KnickS252| |L::I::Yokogawa7651|
+-----------------------+-+ +----+----------+ +----+-------------+
                        |        |                 |
                        |     +--+-----------------+--+
                        |     |Lab::Instrument::Source|
                        |     +---+-------------------+
                        |         |
                     +--+---------+--+
                     |Lab::Instrument|
                     +-------+-------+
                             |
                        +----+----+
                        |Lab::VISA|
                        +----+----+
                             |
                     +-------+-------+
                     |NI-VISA Library|
                     +---+-------+---+
                         |       |
              +----------+-+   +-+----+
              |GPIB Library|   |  OS  |
              +----------+-+   +-+----+
                         |       |
                         |       |Serial connection
          GPIB connection|       |TCP/IP connection
                         |       |USB connection
                         |       |
                +--------+-+   +-+--------+
                |Instrument|   |Instrument|
                +----------+   +----------+
```

Example 1: Use only Lab::VISA

- looks like example programs from the NI VISA manual
  – protocol overhead is pain in the neck. NOT GOOD.

```
#!/usr/bin/perl

use strict;
use Lab::VISA;

# Initialize VISA system and
# Open default resource manager
my ($status,$default_rm)=Lab::VISA::viOpenDefaultRM();
if ($status != $Lab::VISA::VI_SUCCESS) {
    die "Cannot open resource manager: $status";
}

# Open one resource (an instrument)
my $gpib=21;          # we want to open the instrument
my $board=0;          # with GPIB address 21
                      # connected to GPIB board 0 in our computer

my $resource_name=sprintf("GPIB%u::%u::INSTR",$board,$gpib);

($status, my $instr)=Lab::VISA::viOpen(
    $default_rm,       # the resource manager session
    $resource_name,    # a string describing the
    $Lab::VISA::VI_NULL,# access mode (no special mode)
    $Lab::VISA::VI_NULL # time out for open (no time out)
);

if ($status != $Lab::VISA::VI_SUCCESS) {
    die "Cannot open instrument $resource_name. status: $status";
}


(...)
```

```
(...)

# Now we are going to send one command and read the result.

# We send the simple SCPI command "*IDN?" which asks the instrument
# to identify itself. Of course the instrument must support this
# command, in order to make this example work.
my $cmd="*IDN?";
($status, my $write_cnt)=Lab::VISA::viWrite(
    $instr,           # the session identifier
    $cmd,             # the command to send
    length($cmd)      # the length of the command in bytes
);
if ($status != $Lab::VISA::VI_SUCCESS) {
    die "Error while writing: $status";
}

# Now we will read the instruments reply
($status,             # indicates if the operation was successful
 my $result,          # the answer string
 my $read_cnt)=       # the length of the answer in bytes
    Lab::VISA::viRead(
        $instr,       # the session identifier
        300           # read 300 bytes
    );
if ($status != $Lab::VISA::VI_SUCCESS) {
    die "Error while reading: $status";
}
# The result string will be 300 bytes long, but only $read_cnt
# bytes are part of the answer. We cut away the rest.
$result=substr($result,0,$read_cnt);

print $result;
```

Example 2: Use Lab::Instrument

- Much nicer!

```
#!/usr/bin/perl

use strict;
use Lab::Instrument::HP34401A;

my $gpib=21;            # we want to open the instrument
my $board=0;            # with GPIB address 21
                        # connected to GPIB board 0 in our computer

# Create an instrument object
my $hp=new Lab::Instrument::HP34401A($board,$gpib);

# Use the id method to query the instruments ID string
my $result=$hp->id();

print $result;
```

Example 3: My first real measurement!

- How about doing a gate sweep?
- That's easy!

```
#!/usr/bin/perl

use strict;
use Lab::Instrument::HP34401A;
use Lab::Instrument::Yokogawa7651;

# Connect to voltage source
my $yoko=new Lab::Instrument::Yokogawa7651({
   GPIB_address    => 14,
   gate_protect    => 0,
});

# Connect to multimeter
my $hp=new Lab::Instrument::HP34401A(0,21);

# Sweep Yokogawa from 0 to 1 volt (10 steps)

for (my $volt=0; $volt<=1; $volt=$volt+0.1) {
   # set Yokogawa
   $yoko->set_voltage($volt);

   # wait a second
   sleep(1);

   # read multimeter
   my $vmeas=$hp->read_voltage_dc();

   # print values
   print $volt,"\t",$vmeas,"\n";
}
```

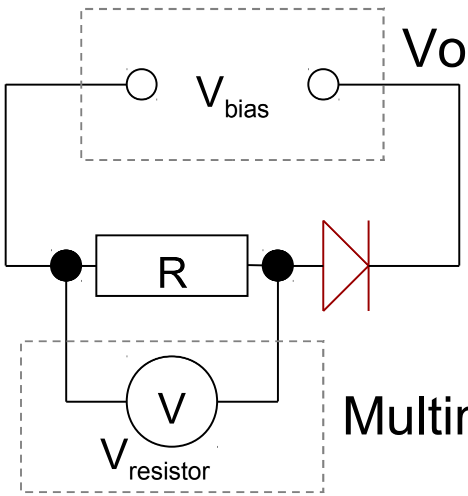
Lab::Tools

- Provides additional tools to write better measurement scripts

- Store metadata alongside data
  - date and time
  - settings of additional instruments
  - ratio of voltage divider
  - color of the shirt you are wearing
  - everything that might be important for a later interpretation of the data

- Don‘t repeat yourself
  - Use above collected information automatically
  - Automatically plot data with correct axes, scaling, labels etc.

Metadata philosophy



Voltage source (Knick)

Multimeter (HP)

Constants

- R = 1000 Ohm

Columns

- C1: V_{bias}
- C2: $V_{resistor}$
- unit
- description…

1.5	+8.19300500E-01
1.4	+7.23413000E-01
1.3	+6.28083900E-01
1.5	+8.19300500E-01

Axes

- Axis “bias voltage” (C1)
- Axis “diode current” (C2 / R)
- Axis “diode resistance” (R * (C1 / C2 – 1))
- unit
- expression
- description…

Plots

- Plot “diode current” axis “diode current” over axis “bias voltage”
- Plot “diode resistance” axis “diode resistance” over axis “bias voltage”
- logscale
- grid
- ranges…

Example 4: Using Lab::Measurement

- metadata, live plot

```
#!/usr/bin/perl

use strict;
use Lab::Instrument::KnickS252;
use Lab::Instrument::HP34401A;
use Lab::Measurement;
use Time::HiRes qw/usleep/;

##################################

my $start_voltage  =  1.5;
my $end_voltage    = -3.5;
my $step           = -0.05;

my $knick_gpib     = 14;
my $hp_gpib        = 21;

my $sample         = "Zenerdiode";
my $title          = "Messung mit Lab::Measurement";
my $comment        = <<COMMENT;
Reihenschaltung aus Widerstand 1kOhm und Zenerdiode.
COMMENT

##################################

my $knick=new Lab::Instrument::KnickS252({
    'GPIB_board'    => 0,
    'GPIB_address'  => $knick_gpib,
    'gate_protect'  => 0,
});

my $hp=new Lab::Instrument::HP34401A(0,$hp_gpib);

my $measurement=new Lab::Measurement(
    sample          => $sample,
    title           => $title,
    filename_base   => 'zener_kennlinie',
    description     => $comment,

    live_plot       => 'diode current',

    constants       => [
        {
            'name'          => 'R',
            'value'         => '1000',
        },
    ],
    columns         => [
        {
            'unit'          => 'V',
            'label'         => 'V_{bias}',
            'description'   => 'Bias Voltage',
        },
        {
            'unit'          => 'V',
            'label'         => 'Amplifier output',
            'description'   => 'Voltage drop on serial resistor',
        }
    ],
    axes            => [
        {
            'unit'          => 'V',
            'expression'    => '$C0',
            'label'         => 'V_{bias}',
            'description'   => 'Bias voltage',
            'min'           => ($start_voltage < $end_voltage)
                               ? $start_voltage
                               : $end_voltage,
            'max'           => ($start_voltage < $end_voltage)
                               ? $end_voltage
                               : $start_voltage,
        },
        {
            'unit'          => 'mA',
            'expression'    => '1000*($C1/R)',
            'label'         => 'I_{diode}',
            'description'   => 'Current through diode',
            'min'           => '-1.2',
            'max'           => '1.2',
        },
        {
            'unit'          => 'Ohm',
            'expression'    => 'R * ($C0/$C1 - 1)',
            'label'         => 'R_{diode}',
            'description'   => 'Diode resistance',
        },
    ],
    plots           => {
        'diode current'     => {
            'type'          => 'line',
            'xaxis'         => 0,
            'yaxis'         => 1,
            'grid'          => 'xtics ytics',
        },
        'diode resistance' => {
            'type'          => 'line',
            'xaxis'         => 0,
            'yaxis'         => 2,
            'grid'          => 'xtics ytics',
            'logscale'      => 'y',
        },
    },
);

$measurement->start_block();

for (
    my $volt = $start_voltage;
    ($volt - $end_voltage) / $step < 0.5;
    $volt += $step
) {
    $knick->set_voltage($volt);
    usleep(500000);
    my $meas = $hp->read_voltage_dc(10,0.0001);
    $measurement->log_line($volt,$meas);
}

my $meta = $measurement->finish_measurement();
```

More utilities: plotter.pl, make_overview.pl

- **plotter.pl**:
  - Reads measurement file, list avaible plots (axes etc)
  - Creates postscript or pdf output
- **make_overview.pl**:
  - Reads all measurements in a directory
  - Generates a postscript or pdf file with plots for each measurement, including the metadata (i.e. constants, parameters, color of your shirt) (via LaTeX)
  - Great for completing your lab book

Other Lab::VISA features

- "**Gate protect**" safety mechanism
  - Makes sure that no voltage is changed to fast
  - Big voltage steps are automatically split into small, slow steps
- **Date/time** handling
  - Date/time column can contain timestamp for every data point
  - Plot data as function of time
- Measurements with **higher dimensionality**
  - Each trace/sweep/line is a "block"
  - Two-dimensional plots, selections of traces, etc.