sigrok:
Adventures in Integrating a Power-Measurement Device

Bartosz Golaszewski,
BayLibre, www.baylibre.com
bgolaszewski@baylibre.com
ACME overview:
(Another Cute Measurement Equipment)
Objectives, key features & status
Problem statement

- Power Management optimization is the key for power-hungry battery-operated devices,
- Limited power measurement equipment,
  - Expensive high-precision lab equipment,
  - Existing low-cost solutions but with limited performances (i.e. accuracy),
  - No standard power measurement connector,
- The community needed a high-perf low-cost standard solution for power measurements.
The answer: ACME Cape
ACME Cape: key features

• Multi-channel:
  – 8, up to 16 with Cape stacking.

• All-in-one solution for power measurement, power control, and temperature measurement.

• Flexible / Evolutive:
  – Extension connector for use with other HW than BBB,
  – New probes can be designed, w/o HW change required on ACME cape.
ACME Cape: key features

- Low-cost,
- Uses TI INA226 & TMP435 components supported by upstream Linux drivers,
- Defines a standard low-cost power measurement connector and provides power probes following this standard,
- Version 2 – USB dongle.
Problem: software support

• Writing our own software-suite
  – costs of development and maintenance
  – duplicating functionalities
  – duplicating bugs
  – clients don't like learning new programs

• Contributing to a well-known and supported project
  – help from the community
  – existing code base and documentation
  – brand appealing to users
Problem: software support

- ACME is open hardware,
- ACME needs a complete open-source software suite,
- Sigrok supports power measurement devices,
- Let’s contribute to sigrok!
sigrok overview:
portable, cross-platform, Free/Libre/Open-Source signal analysis software suite
Design goals and features
sigrok: key features

• flexible,
• cross-platform,
• hardware-independent,
• supports various device types,
• modular architecture.
Broad hardware support

• 129 supported devices
• 20 in progress
• Initially developed for logic analyzers
• Now supports various device types: logic analyzers, oscilloscopes, multimeters, energy meters, sound level meters, thermo-, anemo- and hygrometers, dataloggers & many, many more.
Broad hardware support
Cross-platform

• **Works on:** Linux, Mac OS X, Windows, FreeBSD, OpenBSD, NetBSD, Android.

• **Now available in Buildroot (BayLibre contribution).**
Flexible input/output

• Supports various file formats:
  – binary, analog, ASCII, hex, CSV, gnuplot, VCD, WAV, ChronoVu LA8, OLS.

• Transformation modules (work in progress):
  – Allows transformation of data between the source and output: nop, scale, invert, average, adc/dac (analog to/from digital conversion).

• collectd plugin available
Various frontends

• Command-line: sigrok-cli
• GUI: PulseView
  – Aimed mainly at logic analyzers,
  – Channel grouping support
  – Qt based,
  – Fast $O(\log N)$ signal rendering at all zoom levels.
• sigrok-meter (work-in-progress):
  – Written in Python (2 & 3) + PyQt/PySide,
  – Uses Python bindings generated by SWIG,
  – Aimed at multimeters and dataloggers.
Various frontends - sigrok-cli

Examples:

- sigrok-cli --scan
- sigrok-cli --driver=baylibre-acme --show
- sigrok-cli --driver=baylibre-acme --get probe_factor --channel-group=Probe_1
- sigrok-cli --driver=baylibre-acme --config probe_factor=80 --set --channel-group=Probe_1
- sigrok-cli --driver=baylibre-acme --samples=50 --config samplerate=100
- sigrok-cli --driver=baylibre-acme --time=10s --output-format=analog
- sigrok-cli --driver=baylibre-acme --continuous --transform-module=scale:factor=3.14
Various frontends – PulseView
Various frontends – PulseView

• Android port:
  – Not written from scratch,
  – Portable C++11 + minimal Android ‘glue’,
  – Reuses libsigrok and libsigrokdecode together with all the functionalities (protocol decoders!).
Various frontends – PulseView
Various frontends – sigrok-meter

- Demo device, A0: 25.000000 V
- Demo device, A1: -23.776413 V
- Demo device, A2: -20.000000 V
- Demo device, A3: -10.000000 V
- UNI-T UT61D, P1: 0.000000 Hz
- UNI-T UT61E (UT-D02 cable), P1: 0.001700 V DC
- Voltcraft K204, T1: 24.400000 °C
- Voltcraft K204, T2: 24.000000 °C
- Voltcraft K204, T3: 26.299999 °C
- Voltcraft K204, T4: 24.500000 °C
Protocol decoders

• Way to easily visualize data captured by logic analyzers,
• Written in Python3,
• Stackable,
• Even allow to decode ARM CPU instructions and associate them with code snippets!
Protocol decoders
sigrok architecture

• Reusable libraries:
  – libsigrok, libsigrokdecode.

• Configurable compilation:
  – libftdi, libserialport, libusb, libsigrokdecode.

• Bindings:
  – C++, Python, Java.

• Modular drivers.
sigrok architecture

- sigrok-cli
- PulseView
- sigrok/python bindings
- sigrok-meter
- device firmware
- libsigrokdecode
- libsigrok

Supporting libraries:
- libglib
- libzip
- libserialport
- libusb
-libftdi
### sigrok flow

#### Client
- Sends config requests
- Sets up callbacks
- Starts the acquisition

#### Driver
- Translates client requests
- Sends out acquired data packets

#### libsigrok
- Feed packets to the client
- Separates client from driver logic
sigrok: supporting new hardware
Implementing new libsigrok drivers - tutorial based on ACME
sigrok: supporting new hardware

- **sigrok-util/source/new-driver**
  - updates Makefile.am and configure.ac,
  - adds driver struct to global driver list in src/drivers.c.

- **Implementation split into:**
  - api.c,
  - protocol.h,
  - protocol.c.

- **Goal:**
  - implement device specific callbacks and let the sigrok framework handle the rest.
Callback based driver:

```c
SR_PRIV struct sr_dev_driver baylibre_acme_driver_info = {
    .name = "baylibre-acme",
    .longname = "BayLibre ACME (Another Cute Measurement Equipment)",
    .api_version = 1,
    .init = init,
    .cleanup = cleanup,
    .scan = scan,
    .dev_list = dev_list,
    .dev_clear = dev_clear,
    .config_get = config_get,
    .config_set = config_set,
    .config_list = config_list,
    .dev_open = dev_open,
    .dev_close = dev_close,
    .dev_acquisition_start = dev_acquisition_start,
    .dev_acquisition_stop = dev_acquisition_stop,
    .priv = NULL,
};
```
Device instance

```c
struct sr_dev_inst {
    struct sr_dev_driver *driver;
    int status;
    int inst_type;
    char *vendor;
    char *model;
    char *version;
    char *serial_num;
    char *connection_id;
    GSList *channels;
    GSList *channel_groups;
    void *conn;
    void *priv;
    struct sr_session *session;
};
```
**sigrok:** supporting new hardware

```c
int (*init)(struct sr_context *sr_ctx);
int (*cleanup)(void);
```

Called after the driver is loaded or before it's unloaded.

Helpers available – `std_init()`, `std_dev_clear()` etc.

Very basic init function:

```c
static int init(struct sr_context *sr_ctx)
{
    return std_init(sr_ctx, di, LOG_PREFIX);
}
```
sigrok: supporting new hardware

GSList *(*scan)(GSList *options);

- Initialize and scan for devices.
- Driver should do all the initialization required.
- Return NULL if no device found or the list of struct sr_dev_inst.
sigrok: supporting new hardware

GSList *(*dev_list)(void);
int (*dev_clear)(void);

- Get & clear the list of device instances the driver knows about,
- Usually just:

    static GSList *dev_list(void)
    {
        return ((struct drv_context *)(di->priv))->instances;
    }
Get/set configuration options & list all available values for given option.
sigrok:  supporting new hardware

• Options listed in sr_configkey in libsigrok.h.
• Defined in src/hwdriver.c.
• Reuseable options e.g. ACME shunt resistance -> probe_factor.
• Well-known data types allow for options to be easily understood by GUIs.
sigrok: supporting new hardware

General device options and per-channel-group options, e.g.:

```c
static const uint32_t devopts[] = {
    SR_CONF_CONTINUOUS | SR_CONF_SET,
    SR_CONF_LIMIT_SAMPLES | SR_CONF_GET | SR_CONF_SET,
    SR_CONF_LIMIT_MSEC | SR_CONF_GET | SR_CONF_SET,
    SR_CONF_SAMPLERATE | SR_CONF_GET | SR_CONF_SET | SR_CONF_LIST,
};

static const uint32_t devopts_cg[] = {
    SR_CONF_PROBE_FACTOR | SR_CONF_GET | SR_CONF_SET,
    SR_CONF_POWER_OFF  | SR_CONF_GET | SR_CONF_SET,
};
```
sigrok: supporting new hardware

```c
int (*dev_open)(struct sr_dev_inst *sdi);
int (*dev_close)(struct sr_dev_inst *sdi);
```

Device specific callbacks called before and after starting data acquisition, setting a config option etc.

Several boilerplate reducing helpers available for USB and serial devices:
std_serial_dev_open() etc.
sigrok: supporting new hardware

```c
int (*dev_acquisition_start) (const struct sr_dev_inst *sdi, void *cb_data);
int (*dev_acquisition_stop) (struct sr_dev_inst *sdi, void *cb_data);
```

- Start/stop data acquisition
- Setup callbacks and polling machinery
- `_source_add_*` & `_source_remove_*` functions
sigrok: supporting new hardware

From agilent-dmm/api.c:

```c
static int dev_acquisition_start(const struct sr_dev_inst *sdi, void *cb_data)
{
  (...)

  /* Send header packet to the session bus. */
  std_session_send_df_header(cb_data, LOG_PREFIX);

  /* Poll every 100ms, or whenever some data comes in. */
  serial = sdi->conn;
  serial_source_add(sdi->session, serial, G_IO_IN, 100,
                   agdmm_receive_data, (void *)sdi);

  return SR_OK;
}
```
sigrok: supporting new hardware

• Existing frameworks:
  – USB,
    • USBTMC,
  – Serial,
  – SCPI,
    • VXI-11,
  – gpio (introduced by ACME).

• Most devices have USB or serial connectivity:

• Unusual drivers:
  – ACME,
  – BeagleLogic.
Pitfalls

• Per probe config options,
  – Using --channel-group parameter to set options for a single probe (tried using key-value arguments).
• Proper callback setup in dev_acquisition_start.
Upstreaming effort

• ACME driver for libsigrok, a couple of new features & several bug-fixes merged upstream by BayLibre:
  – Responsive maintainers,
  – Help available on IRC:
    • Fixed an interesting bug in Doxyfile preventing from building libsigrokcxx via buildroot together.
• sigrok packages available in Buildroot.
• Several extensions and bug-fixes for ina2xx and tmp401 drivers in Linux.
ACME & sigrok demo
ACME & sigrok technical showcase today at 6:30 pm

Q & A

Resources:
http://sigrok.org/
http://baylibre.com/acme/
http://sigrok.org/wiki/BayLibre_ACME
Thank You!