Ceylan-Oceanic: Enocean facilities in Erlang

Organisation: Copyright (C) 2022-2022 Olivier Boudeville

Contact: about (dash) oceanic (at) esperide (dot) com

Creation date: Wednesday, September 7, 2022

Lastly updated: Sunday, November 13, 2022

Version: 0.0.1

Status: In development

Dedication: Users and maintainers of the `Ceylan-Oceanic` library.

Abstract: The role of the `Ceylan-Oceanic` library is to provide Erlang-based facilities for the support of the Enocean building automation system.

The latest version of this documentation is to be found at the official Ceylan-Oceanic website (`http://oceanic.esperide.org`).

This documentation is also mirrored here.

Table of Contents

Overview

The Ceylan-Oceanic library provides Erlang-based facilities for the support of the Enocean building automation system, whose devices are generally energy-harvesting / very low-consumption and wireless (supported frequencies around 900 MHz, depending on countries; for a range of up to 300 meters in the open, and up to 30 meters inside buildings).

So Enocean, whose slogan could be "no wire, no battery", is rather unique. No Wifi, no IP connectivity either, hence no real risk of data leak.

Besides Erlang, Ceylan-Oceanic relies only on Ceylan-Myriad and is a rather autonomous part of the Ceylan project. Ceylan-Oceanic can be readily built and run on most Unices, including of course GNU/Linux.

The project repository is located here.

At least a basic knowledge of Erlang is expected in order to use Ceylan-Oceanic.

Purpose

The main motivation of Oceanic is to provide some basic home automation features, especially in terms of security, in order to be able to:

- **intercept and decode telegrams** emitted by sensors, notably single-input contacts (to detect the opening/closing of doors or windows), temperature / humidity sensors or to detect electricity outages, typically to implement one's own alarm center
- **generate and emit telegrams** to control various electrical devices (e.g. lamps), typically to turn on an electric heater or to run one's own presence simulator

Progress & Enocean Coverage

The targeted basic Enocean support has been implemented, so EEP Enocean telegrams can be intercepted and, for the supported EEPs (others may be quite easily added), such telegrams can be properly decoded, as incoming higher-level events.

Reciprocally, telegrams for the supported EEPs can also be encoded and sent.

Yet, for some reason (still unknown), currently our target device (a smart plug) does not seem to accept them (no reaction).

This problem does not seem specific to Oceanic, insofar as recording directly from the command-line an accepted telegram from the USB gateway and sending it exactly as it was (replay) has no effect either on said device - whereas no anti-replay mechanism seems to be enforced here. Investigations are in progress.

Oceanic can also execute a few common commands directly onto the gateway chip, and perform teach-in operations.

Testing Ceylan-Oceanic in Two Steps

Now, let's discuss these subjects a bit more in-depth.

Hardware Prerequisites

In terms of Enocean devices, one needs typically:

- any kind of emitter/sensor, for example a single-input contact/rocker button like these ones
- a receiver, typically a USB gateway, which includes a UART for asynchronous serial communication with an integrated RF module

Popular USB dongles, which often rely on the TCM 310 chip, include the USB300 one (around 37 Euros in France), or the USB310 one (around 50 Euros in France) that we prefer as it features a SMA connector, which allows an external antenna to be connected in order to boost emission / reception ranges inexpensively.

We will rely here on such a configuration.

Operating System Support

Once the USB dongle is connected (here on an Arch Linux host), `lsusb` tells us that it is detected as:

```
Bus 003 Device 009: ID 0403:6001 Future Technology Devices International, Ltd FT232 Se
```

(which applies both to USB300 and USB310)

We will interact with this USB gateway as if it was a serial port.

Rather than having it designated by an obscure, potentially changing name (like `/dev/ttyUSB0`, `/dev/ttyUSB1`, etc.), we prefer assigning it a fixed, well-chosen path, like `/dev/ttyUSBEnOcean`.

For that, one may define a suitable udev rule, typically stored in `/etc/udev/rules.d/99-enocean.rules`, whose content can simply be:

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SYMLINK+="ttyUSBE
```

Following extra option could be added to the previous line, in order to set the group of this TTY: `GROUP="dialout"` or `GROUP="uucp"`, in which case your user shall be in that group (rather than running `sudo chmod 777 /dev/ttyUSB0` each time the device is inserted for example).

So one may prefer:

```
SUBSYSTEM=="tty", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SYMLINK+="ttyUSBE
```

and:

```
$ sudo usermod -a -G uucp my_username
```

One may run `sudo udevadm control --reload-rules && sudo udevadm trigger` to ensure that these changes are taken into account.

Then inserting said dongle should generate log entries that `journalctl -xe` can show, like (timestamps and hostname edited):

```
kernel: usb 3-11: new full-speed USB device number 9 using xhci_hcd
kernel: usb 3-11: New USB device found, idVendor=0403, idProduct=6001, bcdDevice= 6.0
kernel: usb 3-11: New USB device strings: Mfr=1, Product=2, SerialNumber=3
kernel: usb 3-11: Product: FT232R USB UART
kernel: usb 3-11: Manufacturer: FTDI
kernel: usb 3-11: SerialNumber: A600AVJD
mtp-probe[74533]: checking bus 3, device 9: "/sys/devices/pci0000:00/0000:00:14.0/usb3
kernel: ftdi_sio 3-11:1.0: FTDI USB Serial Device converter detected
kernel: usb 3-11: Detected FT232RL
kernel: usb 3-11: FTDI USB Serial Device converter now attached to ttyUSB0
mtp-probe[74533]: bus: 3, device: 9 was not an MTP device
mtp-probe[74548]: checking bus 3, device 9: "/sys/devices/pci0000:00/0000:00:14.0/usb3
mtp-probe[74548]: bus: 3, device: 9 was not an MTP device
```

On insertion we have then, with the former settings:

```
$ ls -l /dev/ttyUSBEnOcean /dev/ttyUSB0
 crw-rw---- 1 root uucp 188, 0 Nov 13 10:24 /dev/ttyUSB0
 lrwxrwxrwx 1 root root      7 Nov 13 10:24 /dev/ttyUSBEnOcean -> ttyUSB0
```

Software Prerequisites

Ceylan-Oceanic relies on general-purpose services offered by Ceylan-Myriad (implying of course Erlang itself), and on an Erlang driver for serial communication.

We use our version[1] of erlang-serial for that, which we prefer installing in user space that way:

```
$ mkdir ~/Software && cd ~/Software
$ git clone https://github.com/Olivier-Boudeville/erlang-serial
$ cd erlang-serial
$ make && DESTDIR=. make install
```

Then using `erlang-serial` will be just a matter of adding it to one's code path.

One may update the `Erlang-serial section` in Oceanic's GNUmakevars.inc in order to take into account any other path convention.

One may run, from the root of Oceanic, `make info-serial` to check that `ERLANG_SERIAL_BASE` points indeed to a directory containing erlang-serial's `ebin` directory.

To test it (whether or not any dongle is connected):

```
$ erl -pa $HOME/Software/erlang-serial/erlang/lib/serial-1.1/ebin
```

[1] A fork of the original erlang-serial, which had to be modified notably in terms of disabled RTS/CTS flow control, in order to be able to properly send data to the Enocean gateway.

```
Erlang/OTP 25 [erts-13.0] [source] [64-bit] [smp:8:8] [ds:8:8:10] [async-threads:1] [

Eshell V13.0  (abort with ^G)
1> serial:start().
<0.82.0>
```

Perfect!

Testing Enocean

Ensure that none of the next serial tools / terminals is left running, otherwise this may block your ability to send telegrams thanks to Oceanic.

To check, one mat rely on:

```
$ lsof /dev/ttyUSBEnOcean
COMMAND    PID      USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
serial  214977 your_user   3u   CHR  188,0      0t0 1066 /dev/ttyUSB0
```

Basic, Direct Command-line Testing

It is as simple as executing from the command-line (thus without Oceanic or Erlang being involved):

```
$ od -x < /dev/ttyUSBEnOcean
0000000 0055 0707 7a01 10f6 2e00 96e1 0130 ffff
0000020 ffff 0039 554b 0700 0107 f67a 0000 e12e
```

(of course for such a binary content to be received, Enocean telegrams must be emitted; the simplest approach is to trigger any Enocean device able to send on demand such telegrams, like a button/rocker/switch)

`hexdump` can be also used to intercept telegrams. If needing to set the transmission speed beforehand, use `stty -F /dev/ttyUSBEnOcean 57600`.

Incoming data can also be recorded and "replayed" (yet this is not expected to activate an Enocean receiver, see Protocol Information):

```
$ cat < /dev/ttyUSBEnOcean > my_record.bin
$ cat my_record.bin > /dev/ttyUSBEnOcean
```

With a Graphical Serial Terminal

One may use cutecom to directly test input/output telegrams.

A priori neither RTS nor DTR shall be enabled, yet in our tests these had no impact.

We recommend using the `Hex` input and output.

Oceanic Testing

From the root of the Ceylan-Oceanic clone, supposing that Myriad and erlang-serial are already available and built:

```
# Ensure that Ceylan-Oceanic is built:
$ make all

$ cd test
$ make oceanic_run

       Running unitary test oceanic_run (third form) from oceanic_test

--> Testing module oceanic_test.

Starting the Enocean test based on the gateway TTY '/dev/ttyUSBEnOcean'.
[debug] Using TTY '/dev/ttyUSBEnOcean' to connect to Enocean gateway, corresponding to
[debug] Stopping serial server <0.84.0>.

--> Successful end of test.

(test finished, interpreter halted)
(command success reported)
```

Enocean Documentation

- [ETS]: Enocean Technical Specifications, notably for:
  - [EEP-gen]: EnOcean Equipment Profiles (e.g. version 3.1.4, 36 pages), a short, general view onto the structure of the various telegram types that are available (e.g. the RPS one)
  - [EEP-spec]: EEP Specification (e.g. version 2.6.7, 270 pages), for a detailed specification of the various equipment profiles (e.g. `F6-01-*` being for *Switch Buttons*)
- [ESP3]: Enocean Serial Protocol (ESP3) - SPECIFICATION (e.g. version 1.51, 116 pages), a point-to-point packet-based protocol that is lower-level in the network stack; of lesser interest here)

Note also that, despite the availability of ERP2 specifications, at least most devices rely on ERP1 ones.

Protocol Information

Provided that the serial link is properly configured (in terms of speed, parity, start/stop bits, RTS/CTS flow control, etc.), it looked as if replaying Enocean telegrams would be pretty straightforward.

However, in practice, if just intercepting a telegram and re-emitting it will be successfully acknowledged in terms of code returned by the target device, it will *not* trigger the intended effect (e.g. the smart plug will not switch on).

Not only we actually receive information different from what was sent (e.g. the dbM measure, the repeating count, etc. are visibly set between the emission and the receiving), but, more importantly, even if forging an apparently acceptable telegram (i.e. creating a correct one, mentioning the EURID of a legit emitter already paired to the receiving device, with proper send-related information), this will still not be sufficient.

We interpret that as the devices relying on a lower-level protocol (possibly. ESP3) than the one that can be handled programmatically (e.g. ERP1 and siblings); a bit like if one was spoofing IP addresses in forged packets whereas the device compares MAC addresses.

As these operations seem to be done through the firmware of the USB gateway, spoofing Enocean traffic may be out of the reach of programs relying on "standard" USB gateways, explaning why Oceanic has to support a whole teach-in procedure.

Support

Bugs, questions, remarks, patches, requests for enhancements, etc. are to be reported to the project interface (typically issues) or directly at the email address mentioned at the beginning of this document.

Additional Information

- EnOcean in Practice (very clear information, in French)

Related Projects

They may be used as sources of inspiration:

- [PY-EN] the rather complete Python EnOcean library, including for its EEP (XML) information
- a Java implementation: enocean4j
- the (Java) OpenEnocean openHAB binding
- a first Rust implementation

Please React!

If you have information more detailed or more recent than those presented in this document, if you noticed errors, neglects or points insufficiently discussed, drop us a line! (for that, follow the Support guidelines).

Ending Word

Have fun with Ceylan-Oceanic!