Vehicle Signal Manager

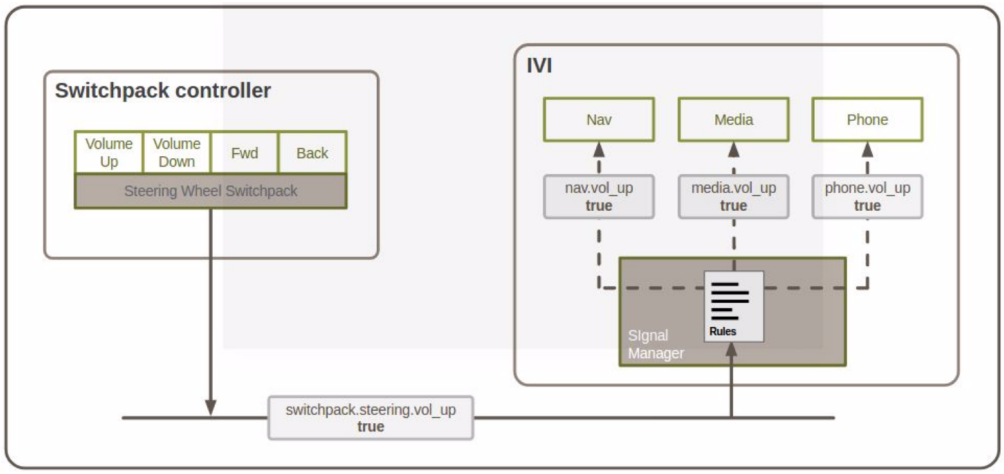
Overview

The Vehicle Signal Manager project is designed create a prototype implementation which abstracts events (such as the transmission changing to "reverse") and possible system reactions (such as the "reverse camera" feed appearing on the display). VSM allows the rules for these relationships to be specified in a configuration file which may be easily maintained by the manufacturer rather than hard-coding them in code as has traditionally been the case.

Features

The prototype vehicle signal manager (`vsm`) currently supports:

- Basic VSM state tracking and signal emission based on VSM rule files and user input
- VSM rule microformat contained within the YAML format
- Setting initial state in the rule files
- The `condition` keyword along with boolean expressions consisting of signal names and literal values, such as `phone_call == 'active'`
- Redirecting verbose output to a separate log file with the `--log-file` flag
- Delayed signal emission with the `delay` keyword
- Pluggable IPC infrastructure and a ZeroMQ IPC module as a proof-of-concept for network IPC
- Detailed logging, such as a timestamp on each line which will allow for later support to "replay" log files for debugging purposes
- Test suite (`tests.py`) which we keep current with every commit



Latest Changes

- Logging within `vsm` is handled by a separate process to avoid blocking the policy engine by disk writes
- By default, `vsm` outputs each rule condition check and its outcome.
    - This can be disabled with the `--no-log-condition-checks` flag
- `vsm` non-essential output (like state dumps) can be redirected to a separate log file with the `--log-file` flag
- `vsm` prints signal emissions and the full VSM state each time any member changes
- initial `vsm` test suite (`tests.py`)
- initial `vsm` implementation

Where to get the software

Development happens at: vehicle_signal_manager repository

Installation and Dependencies

The project can be run in-place and doesn't need to be installed. The requirements are:

- Python 3
- PyYAML
- pyzmq (for the ZeroMQ IPC module)

Running

Run the VSM prototype with a given rule file like:

```
`./vsm sample_rules/simple0.yaml`
```

State changes can be entered at `stdin` like:

```
`phone_call = 'active'`
```

By default, `vsm` will print any resulting signals to `stdout` and log additional details to the log file (default: `vsm.log`).