Vehicle Signal Manager [VSM]

May 11, 2017 | Project Overview

Rudolf J Streif

Expert Group Lead, GENIVI Alliance

Which feature gets the "Volume Up" button press?

GENIVI®

Vehicle state dictates use cases

"Volume Up" pressed:

- **Are we in reverse?**
  Increase Parking Assist volume.

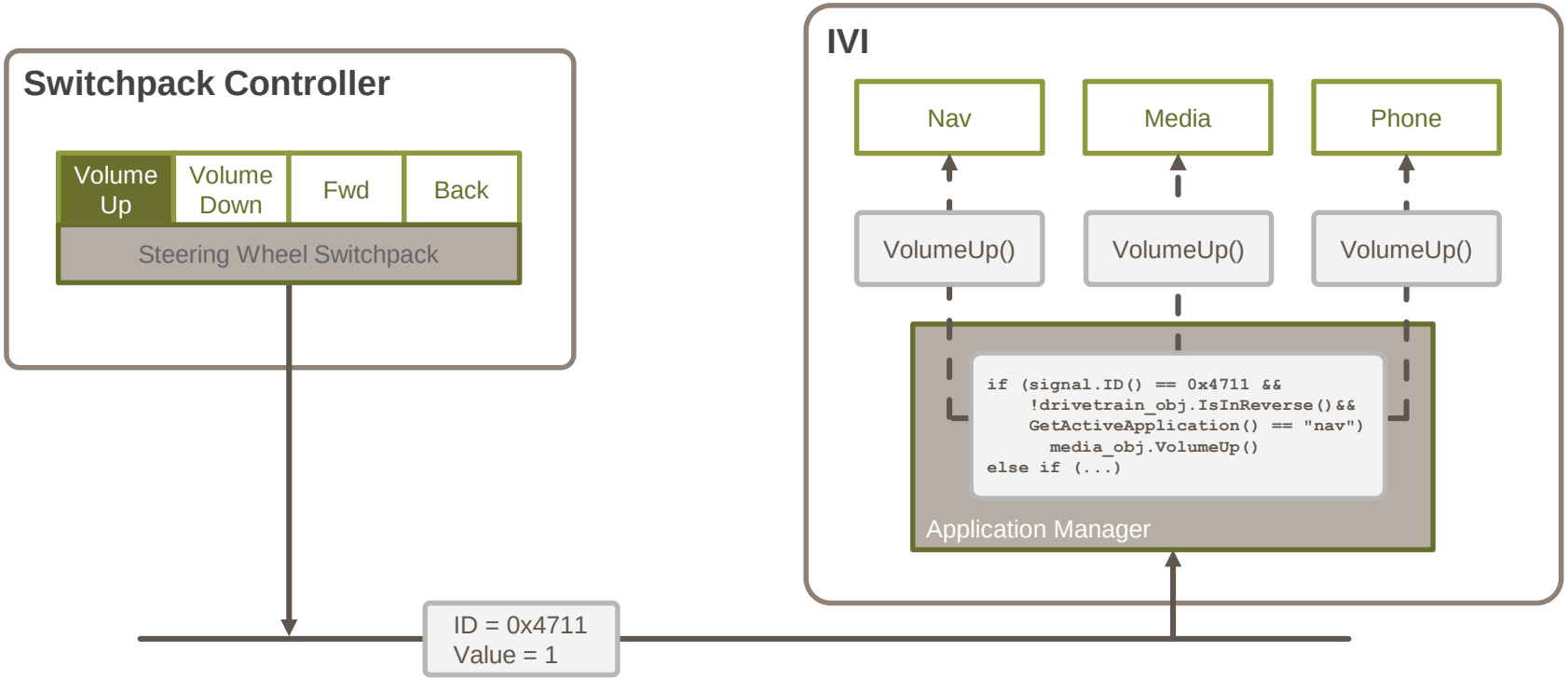
- **Is Navigation active**
  Increase Nav Volume.

- **Are we in an ongoing phone call?**
  Increase Phone Volume.

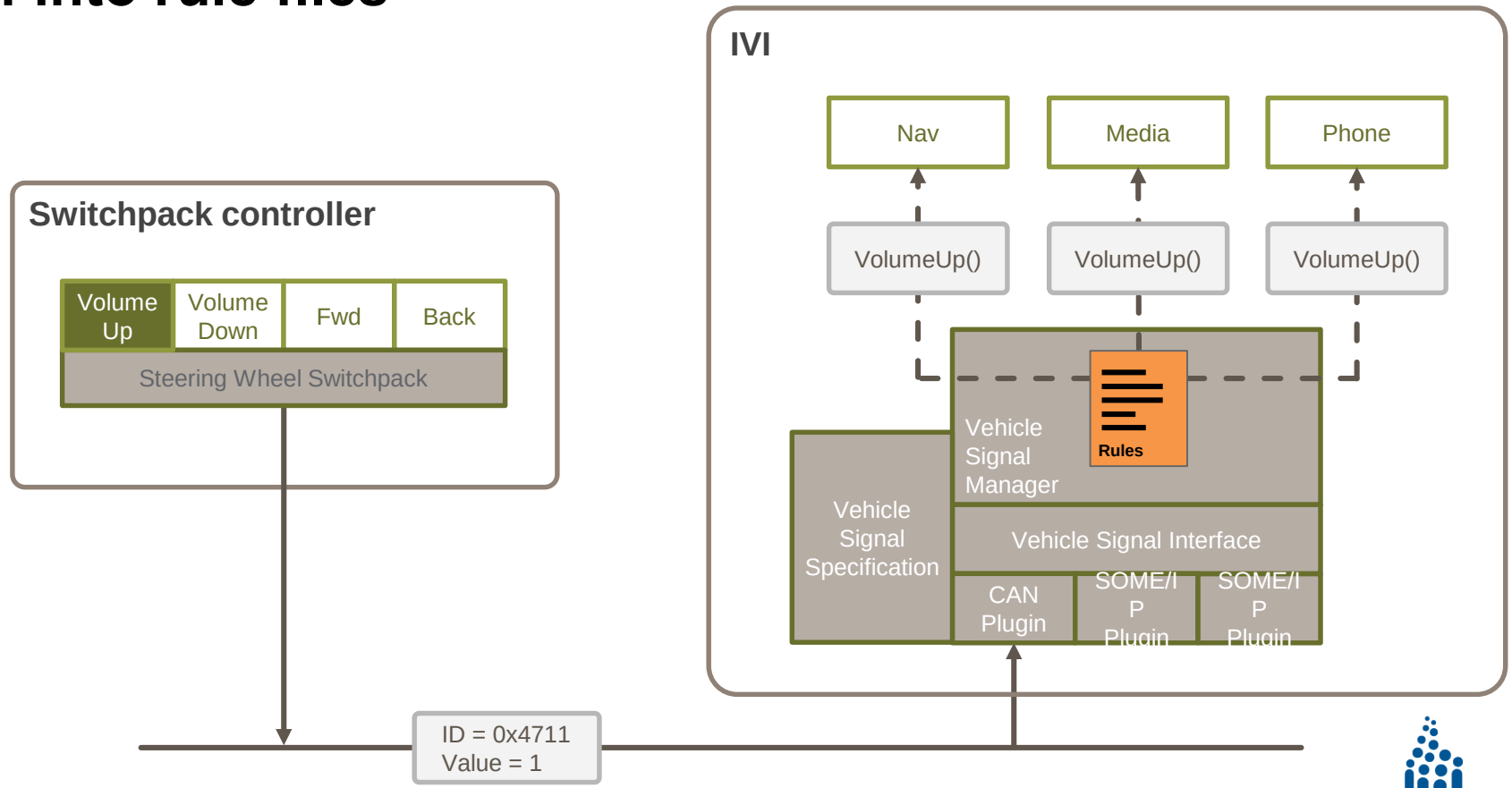
- **None of the above?**
  Increase Media Volume.

GENIVI®

We need to move signal logic out of the code...



… into rule files



Benefits

- **Manage variance complexity**
  One system to handle use cases across vehicle lines, configuration, and regional legislation.

- **Provides testing framework**
  The rule specification driving the signal distribution can also be used to validate signal sequences and timing during test.

- **Updateable via OTA**
  Updated rule specifications can be pushed over the air, without the complexities of SOTA, to adjust fleet behavior.

Project Goals and Objectives

- **Explore signal transformation space**
  Can signal transformation be used to extract call flow logic from features?

- **Understand boundaries between state machines and features**
  What do we encode in signal manager and what stays in the feature?

- **Prepare for production-level implementation**
  Lessons learned fed into potential production variant of the manager.

What is a VSM rule?

- **Monitors vehicle signals for specific conditions ...**

```
- condition: transmission.gear == 'reverse'
```

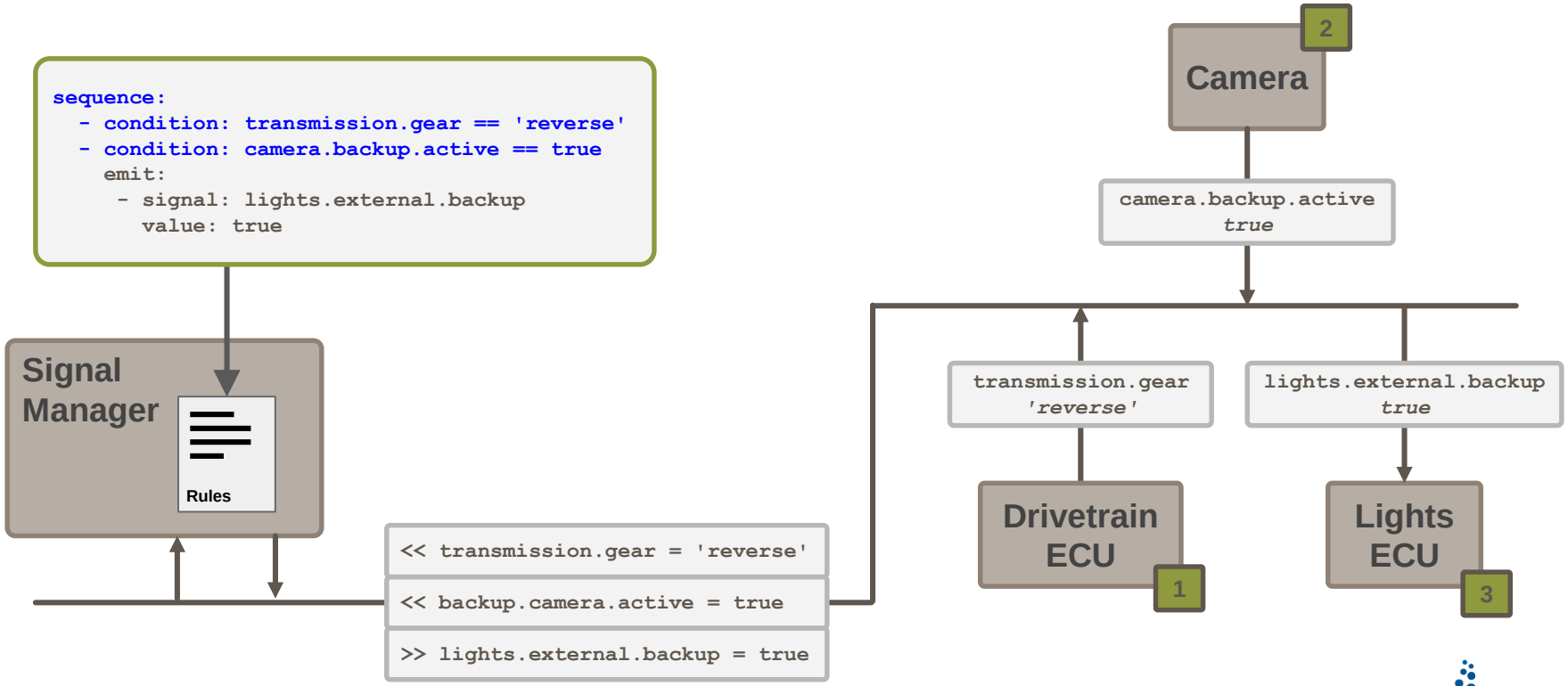
- **… and then emits signals or makes API-calls on its own**

```
  emit:
    signal: lights.external.backup
    value: true
```

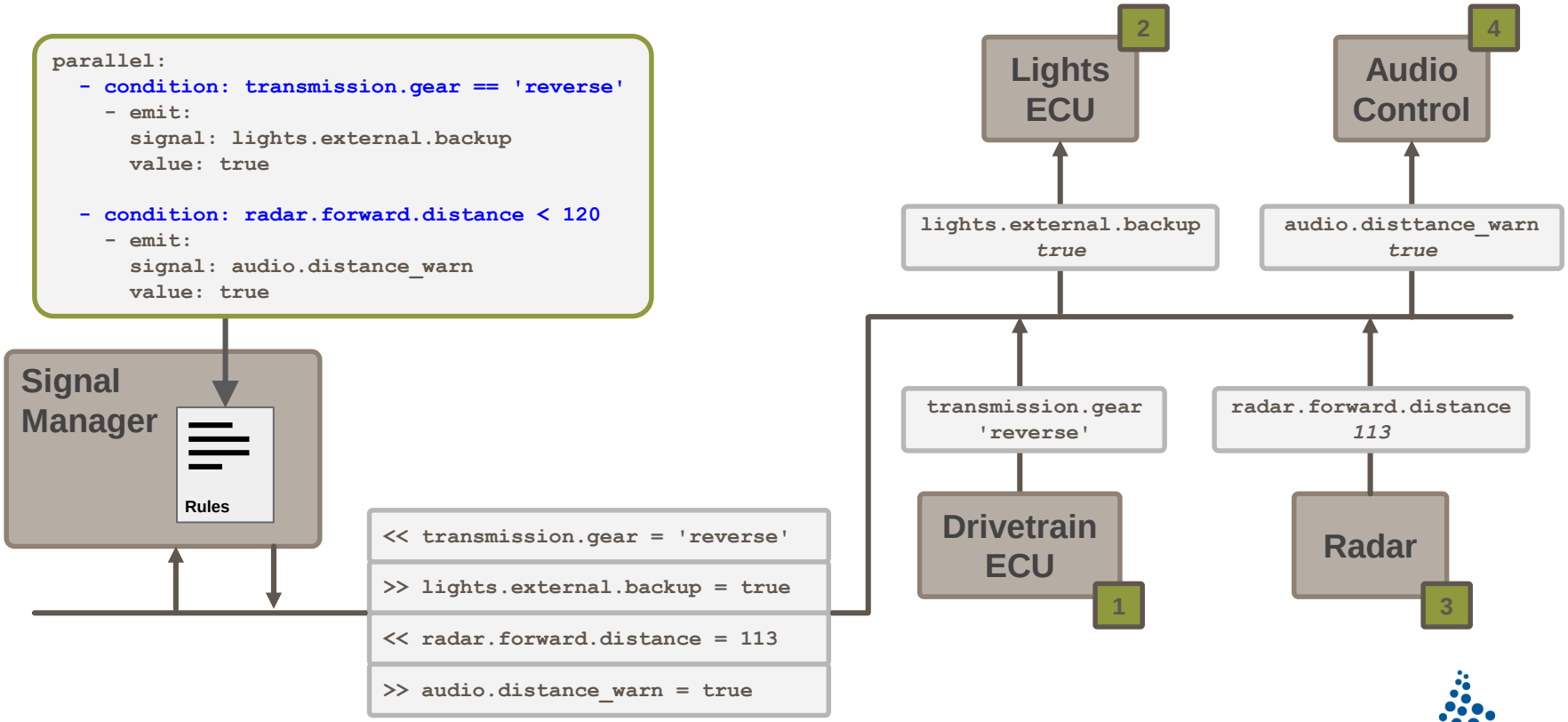
- **All encoded as YAML**

GENIVI®

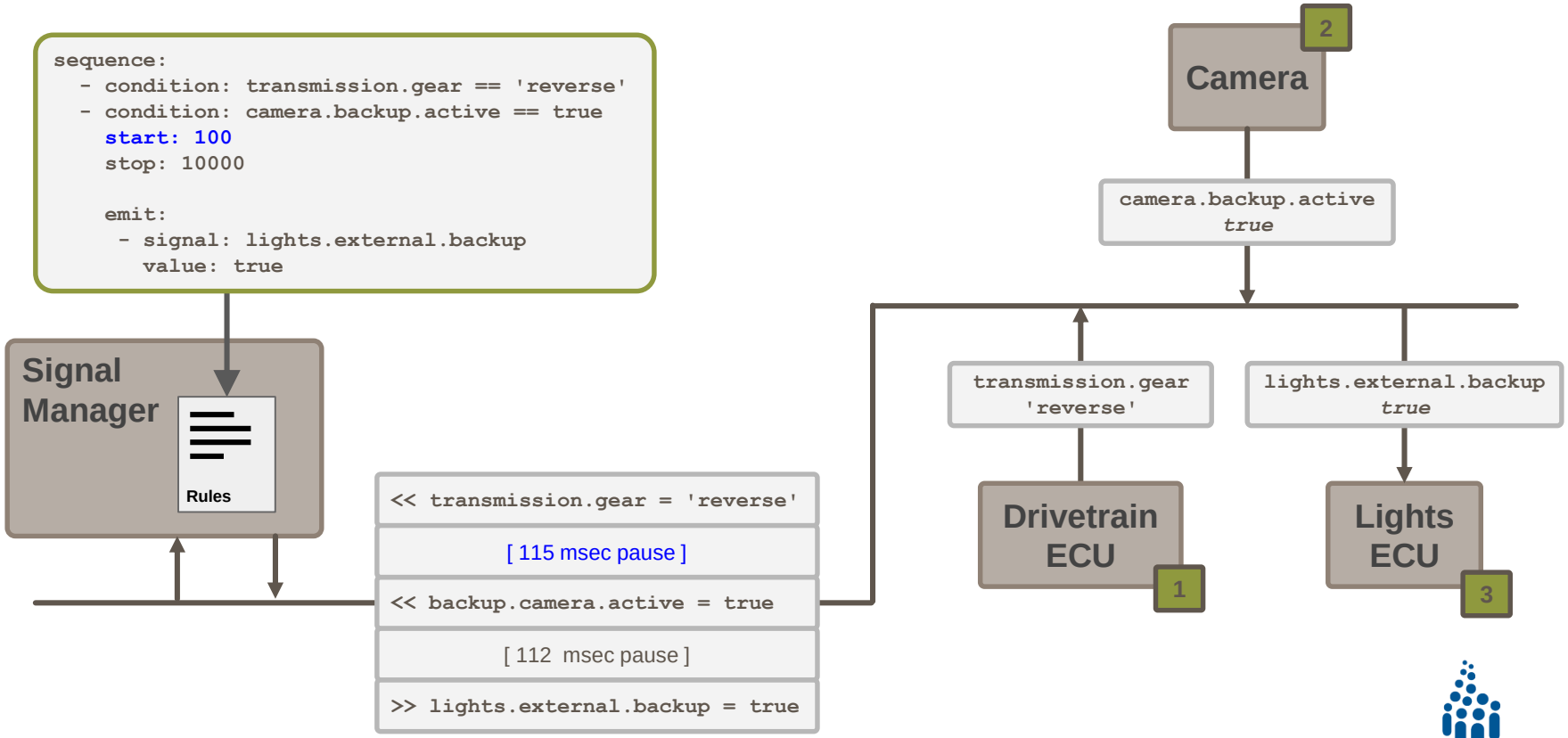
Conditions can be monitored in sequence ...



… Or in parallel

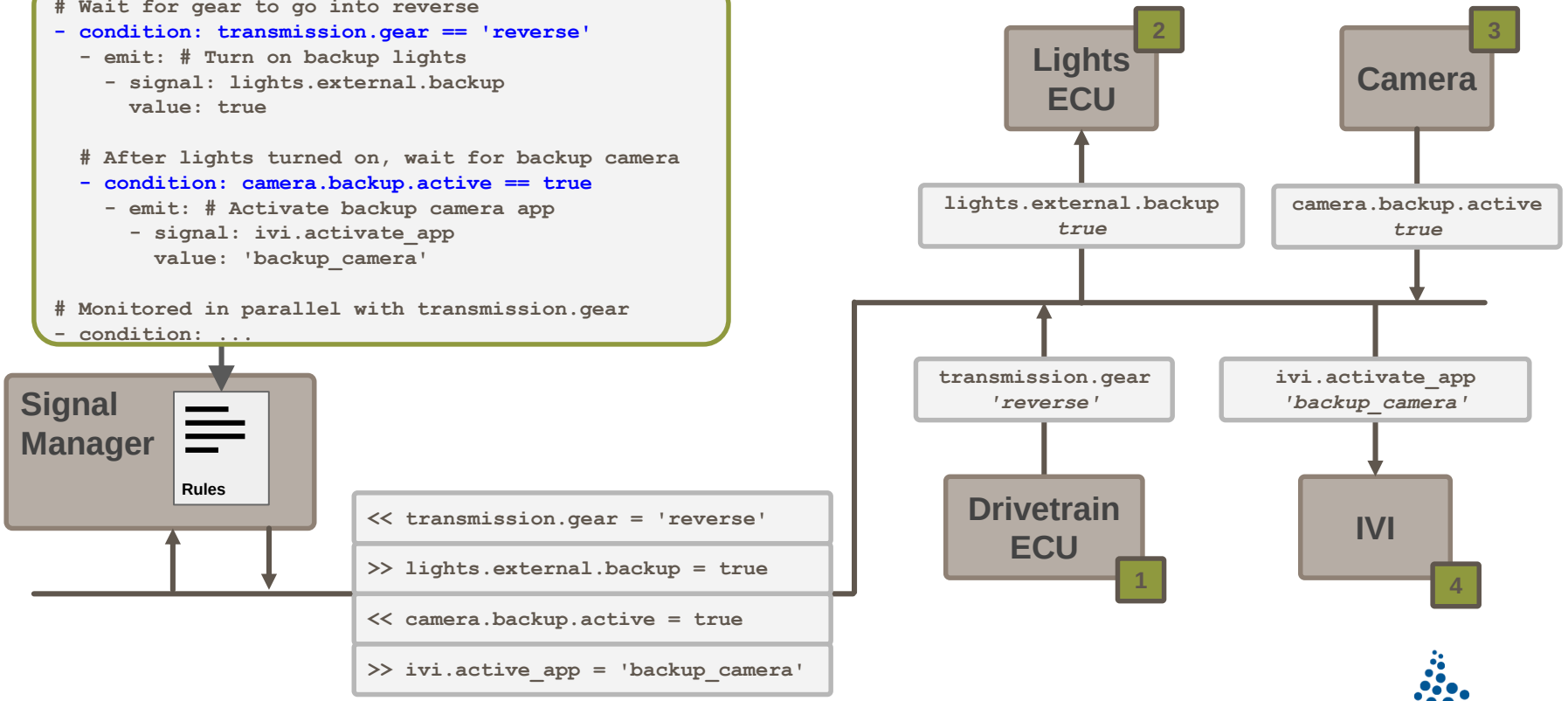


Add timing to get a test specification



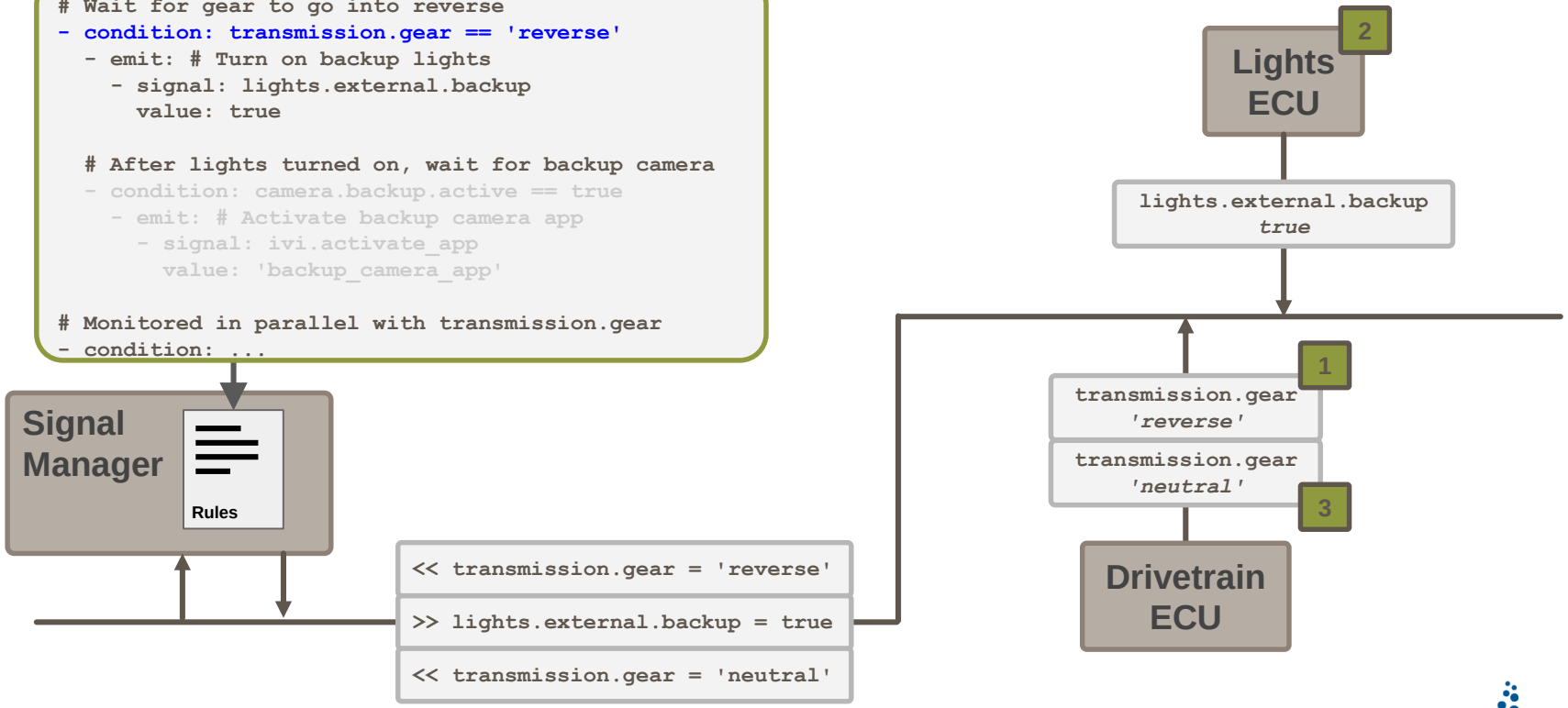
Conditions can be nested...

```
# Wait for gear to go into reverse
- condition: transmission.gear == 'reverse'
  - emit: # Turn on backup lights
    - signal: lights.external.backup
      value: true

  # After lights turned on, wait for backup camera
  - condition: camera.backup.active == true
    - emit: # Activate backup camera app
      - signal: ivi.activate_app
        value: 'backup_camera'

# Monitored in parallel with transmission.gear
- condition: ...
```



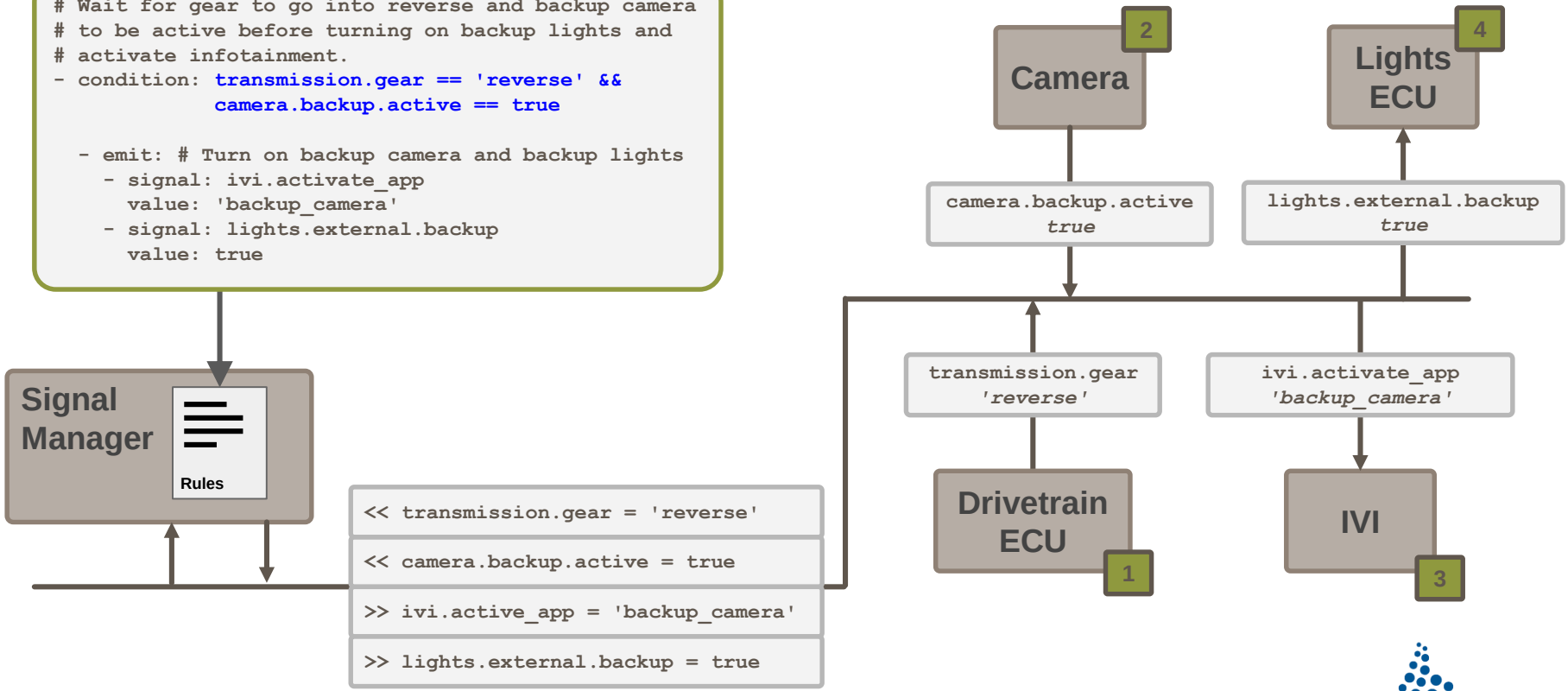
… and cancelled if parent condition turns false

```
# Wait for gear to go into reverse
- condition: transmission.gear == 'reverse'
  - emit: # Turn on backup lights
    - signal: lights.external.backup
      value: true

  # After lights turned on, wait for backup camera
  - condition: camera.backup.active == true
    - emit: # Activate backup camera app
      - signal: ivi.activate_app
        value: 'backup_camera_app'

# Monitored in parallel with transmission.gear
- condition: ...
```



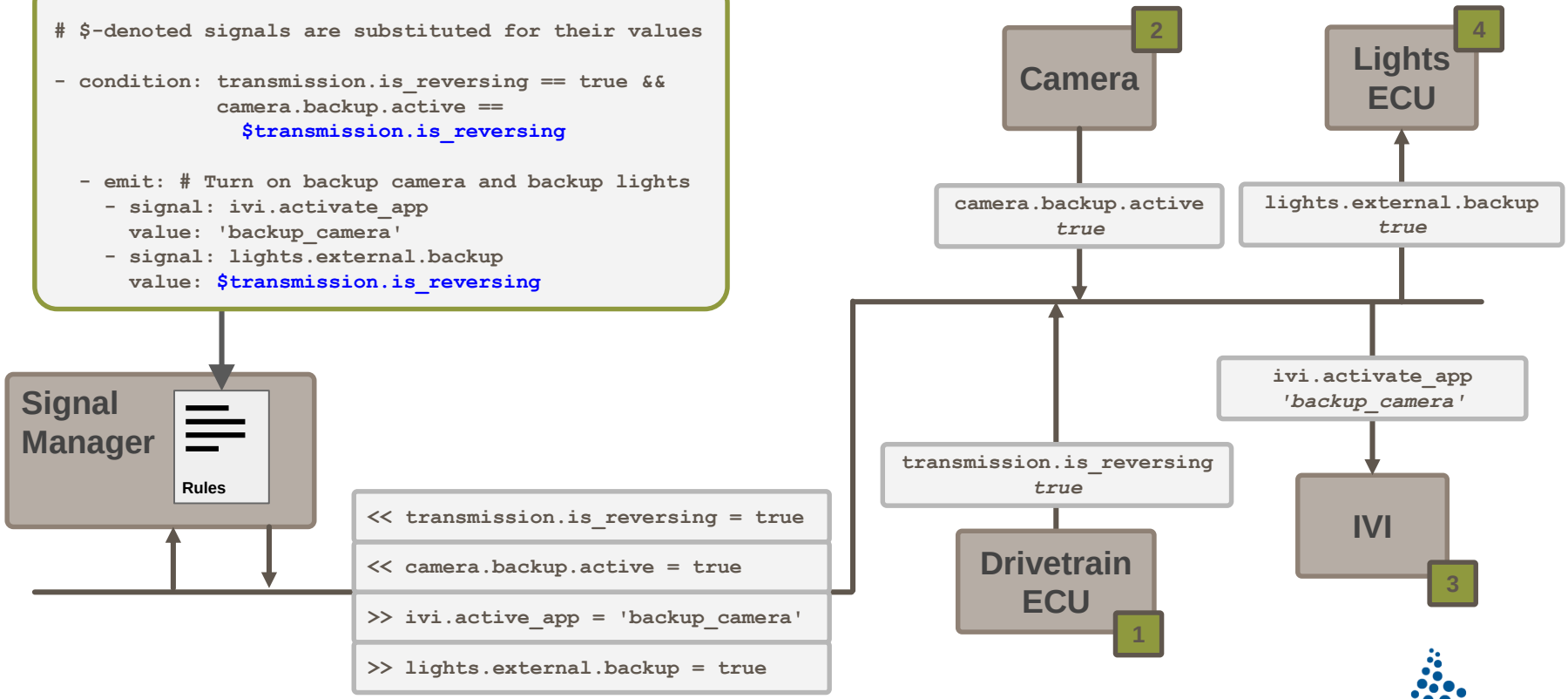
Conditions can be arbitrary expressions...

```
# Wait for gear to go into reverse and backup camera
# to be active before turning on backup lights and
# activate infotainment.
- condition: transmission.gear == 'reverse' &&
             camera.backup.active == true

  - emit: # Turn on backup camera and backup lights
    - signal: ivi.activate_app
      value: 'backup_camera'
    - signal: lights.external.backup
      value: true
```

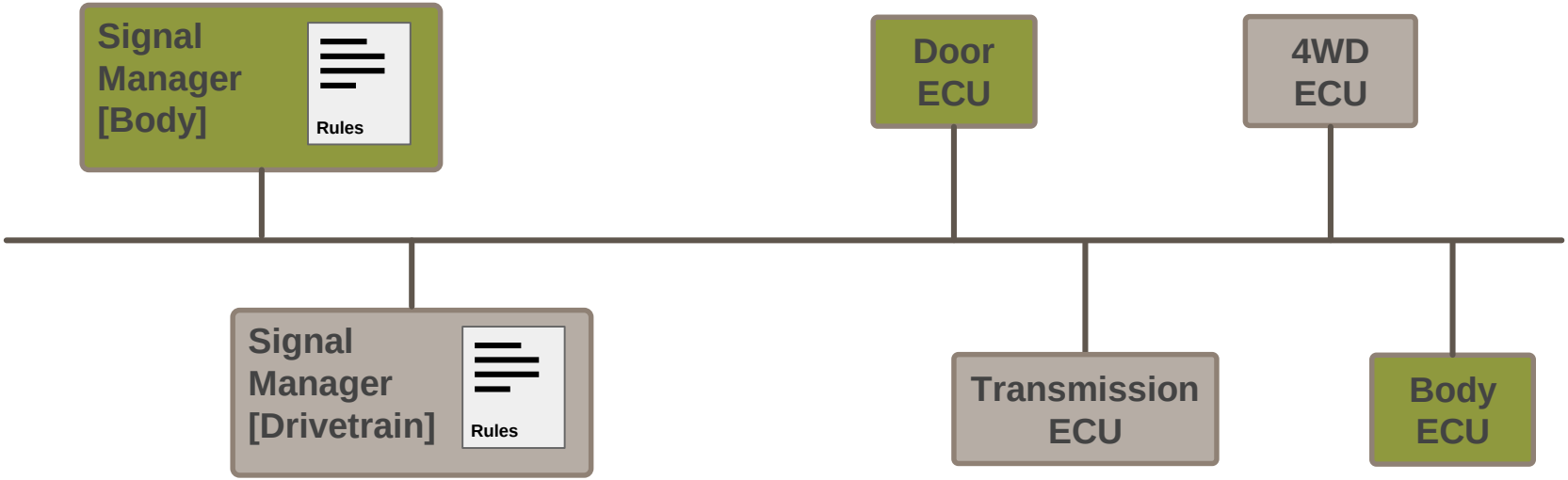


… that can include signal values

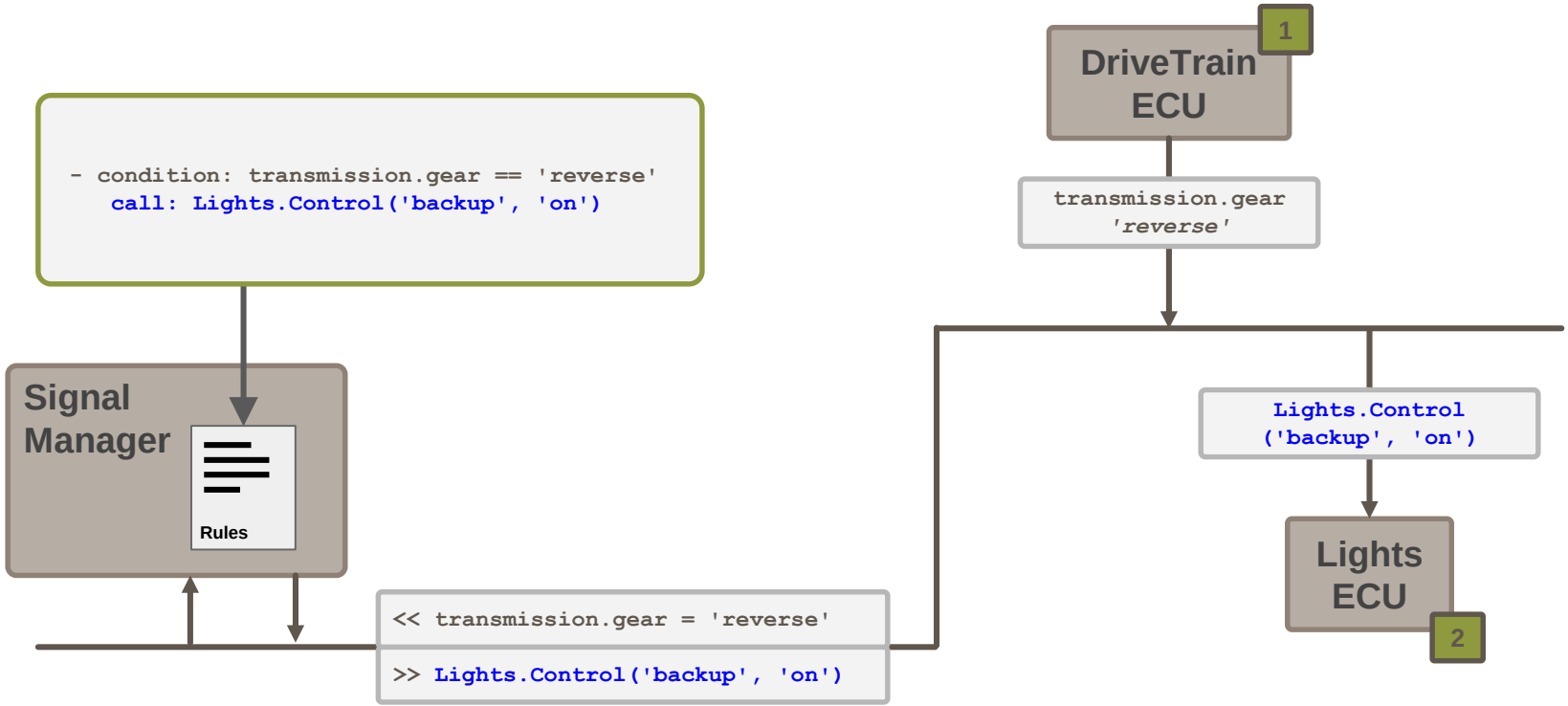
```
# $-denoted signals are substituted for their values

- condition: transmission.is_reversing == true &&
             camera.backup.active ==
               $transmission.is_reversing

  - emit: # Turn on backup camera and backup lights
    - signal: ivi.activate_app
      value: 'backup_camera'
    - signal: lights.external.backup
      value: $transmission.is_reversing
```



You can run multiple signal managers in parallel



Future: API calls?



Conclusion

- **Separate out call flow logic to manage variance complexity**
  Separate rule YAML files for different models, markets, and configuration.

- **Rules are testable in the rig and in the field**
  Timing information can be left in production to detect issues in deployed fleet.

- **Simplified OTA**
  Rules can be pushed over the air without the full validation and installation process required by a software update.

GENIVI®

Thank you!

Rudi Streif, Jaguar Land Rover - rstreif@jaguarlandrover.com
Magnus Feuer, Jaguar Land Rover - mfeuer1@jaguarlandrover.com

Project: https://github.com/genivi/vehicle_signal_manager
Visit GENIVI at http://www.genivi.org or http://projects.genivi.org
Contact us: help@genivi.org