DOMIN

Software Engineer

Digital / Active Suspension

Interview Task

Overview

This document describes a technical task, which is the second stage in our interview process. Through the task, we want to assess your general software design/development and data analysis skills.

Please complete Task 1 AND Task 2

We believe there's enough information in this document for you to complete the tasks. However, please do get in touch if you have any questions.

Please send us your task submission via email no later than 4 days from the date of receipt of this document.

Task 1 – Design a Streaming Service

Context:

This is an open ended system architecture design task; so please treat it accordingly.

Design a streaming service for testing active suspension on a development road vehicle. This system will be operated by test engineers and stream data off the vehicle for use by the Domin engineers working on the project.

The system should consider 4 suspension units, 1 suspension controller and the vehicle. Refer Appendix A for a sample schematic.

Your design should interface with each of the above and capture the data they are producing. The maximum data rate of each of these devices is 1000 Hz and the maximum number of channels is 32 per device. Data types can be floats, integers or strings.

Your system should be capable of storing data offline in the case that internet connectivity is not available (up to 1 hr) and once it is, it should stream the data to the cloud. Local data should also be accessible to the in-vehicle UI via and API for the last 100 seconds.

Once the data is in the cloud it needs be accessible to engineers and analysis processes that will be run on the data.

- An API should be made available that allows engineers to query specific channels, specific time windows or a combination of both.
- Data analysis/processing jobs that detect events in the data streams (such as large bumps in the road, the vehicle doing a left turn at 30 kph). Identified events should be stored for later use and should be available from querying.

Some more considerations.

Over time we will develop standard data analysis scripts that we will want to run. These will enable Domin Engineers find events in their data quickly and efficiently. Examples of this could be taking a corner at 0.2G while a pothole event is encountered (tyre accelerated downwards at 0.9G or greater). How would the system handle these data processing jobs and how would we go back through all the historical data and do this processing.

Other considerations include but are not limited to: How will the data be stored in the long term. How does the system remain responsive whilst being cost effective.

How do you effectively query data over an hour long drive. Consider how you would serve a UI that that needs a low-resolution summary, then loads more and more detail as the user zooms in on events.

Expected outcome: A presentation containing the following details and/or any additional information that you think would be useful.

- Produce an architecture diagram of the above system
- Suggest technologies that you would use

Task 2 – Implement part of the system

Implement **ONE** of the following parts of this system in Python (or a language you are familiar with). Add an additional functionality you see fit from Task 1.

(1) A local data handler that handles data ingestion from local devices (sensors e.g. accelerometers, potentiometers) and a basic web UI for diagnostics . It should be:
- Extensible to work with many devices
- Failure tolerant (no internet, device malfunction)
- Be capable of serving a basic local diagnostics UI, showing latest value for all captured data attributes

OR

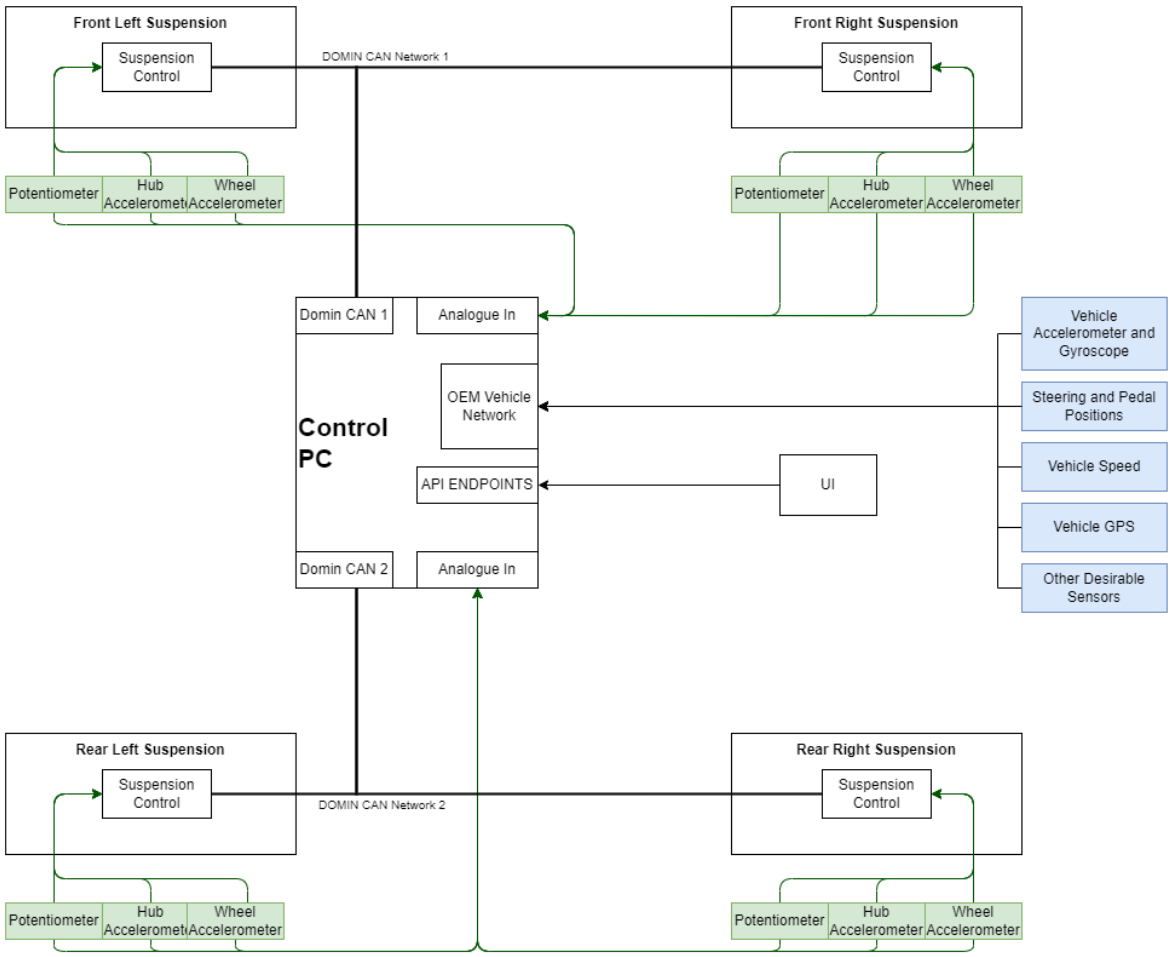
(2) A data processor that can analyse data your system has collected. In the example data provided a vehicle is driven over a standard road bump. How would you analyse the data to find this and log this event in a way that can be used by the rest of your system.
The general case for this would normally mean looking for a certain condition in a specific set of data streams. It should be capable of:
- Finding specific patterns in the data
- Highlighting and visualising patterns found
- Capable of storing the output event
- Capable of ingesting previously identified events (pipelining) for more complex pattern finding.
- Capable of being run in real time so that data is streamed and processed in real time.

Depending on which problem you tackle you may want to mock the inputs you expect. We have provided some CSV and JSON data, along with some Python scripts data that you can use to mock your solution. You can find them on this GitHub repository: Domin-co/domin-swe-interview-task (github.com)

Expected outcome: GitHub repository with your solution, containing the code, any unit tests and instructions for setup and use.

Appendix A

System Architecture



Notes

In this example system Domin are developing a telemetry system to ingest the data from all of the sources in the figure above. The code to do with will live on the Control PC above.

The system will also need a way to serve a UI so the end users of the system can visualise the data live. Here we will need to provide endpoints that enable this.

This diagram shows 4 suspension units. Given Domin Suspension is based around their world leading hydraulics, each suspension unit will be transmitting information about the state of the hydraulic system in each suspension unit (pressure, temperature, suspension position, etc)

OEM in this case is the vehicle we are integrating with. This could be anyone form Toyota to Tesla.