Concordia University | Department of Electrical and Computer Engineering

Project Description

COEN 320 – Introduction to Real-Time Systems (3.0 credits) | Fall 2020

1. Introduction

The project consists of the implementation of a smart system for real-time monitoring and diagnostic of a vehicle's health. It will monitor the status of various vehicle components, in a real-time manner, provide the driver with enhanced driving-related information, and report any malfunctioning or failure in the monitoring vehicle sub-systems. One example of such a system in the market is the ZUS Smart Driving Assistant App, developed by nonda (https://obdapp.nonda.co). The idea is to implement a set of real-time periodic tasks that will frequently obtain data from in-car sensors and display information to the driver.

Vehicles have a built-in self-diagnostic system, in which sensors placed at multiple in-car subsystems (e.g., engine control unit (ECU)) monitor key variables, log data, and report to the driver any malfunctioning component, usually through indicator light on the vehicle's dashboard. There are some applications on the market (e.g., ZUS Smart Vehicle Health Monitor) that obtain data gathered from in-car sensors and provide detailed information to the driver, in a more useful, readable, and suitable way. One of the challenges involved in the development of such applications is the periodic collection of data produced by in-car sensors. The development of real-time tasks to periodically request data from in-car sensors contributes to overcoming the mentioned challenge, while the ODB-II standard can be used from the real-time periodic tasks to obtain diagnostic data from several in-car components. The ODB-II standard specifies the type of diagnostic connector and its pinout, the electrical signaling protocols available, and the messaging format. More details about the ODB-II are presented in the documents posted on Moodle.

In addition to the diagnostic systems, data acquired by in-car sensors and obtained by using the ODB-II protocols are boosting other applications. For instance, it has been used to characterize driver behavior to identify the driver, i.e., the car owner, of a vehicle, and reduce auto-theft attacks (https://www.iit.cnr.it/sites/default/files/human-behavior-characterization.pdf). It can also be used to determine driver style and support car insurance pricing accordingly. Moreover, AI-based applications using vehicular off-board and on-board data, which can be obtained by using the ODB-II protocol, have been designed for vehicle predictive maintenance.

2. Requirements

The objective of this project is to implement a real-time application for the real-time monitoring of a vehicle. **The project consists of the implementation of a set of real-time tasks that will periodically obtain the vehicle's data and display it to the driver.** Ideally, the real-time application to be implemented would need to use the ODB-II to acquire data from the vehicle's

on-board sensors in a real-time manner. Due to the current situation, a dataset with data obtained from different on-board sensors will be provided. The dataset will be available on Moodle. It provides data from different sensors collected every 1 second. More specifically, the project consists of the implementation of the following:

- **Data producer threads:** One real-time, periodic thread for each variable of interest (e.g., vehicle speed, engine speed, and full consumption) to periodically request data gathered from the sensor in charge of it. Again, the ideal scenario would be to use the ODB-II protocol to do this request. However, a dataset will be provided. It means that each periodic thread will need to read the given data stored in the dataset.
- **Data consumer thread:** One real-time thread that will be in charge of updating displayed information to the driver. This thread will need to execute periodically and communicate with data producer threads to obtain needed data and provide obtained information to the driver.

The below table shows the variable of interest and the periodicity that shall be used.

Variable	Periodicity
Full Consumption	10 ms
Engine Speed (RPM)	500 ms
Engine Coolant Temperature	2 s
Current Gear	100 ms
Transmission Oil Temperature	5 s
Vehicle Speed	100 ms
Acceleration Speed Longitudinal	150 ms
Indication of break switch	100 ms

The project should be done in teams of 3 students, implemented in C, and must run in the QNX Neutrino real-time operating system. A preliminary report, final report, and code must be submitted on Moodle by the due dates. The report must clearly state all the assumptions and design decisions made by the team. Moreover, it must clearly describe the technical contributions of every member of the group regarding the design and implementation of the project). A demo will be held during Week 13 of this fall term. During the demo, all the members of the group should attend and be ready to answer questions from the instructor and TA. During the demo, we may also go through the code and the report.

3. Deliverables:

The deliverables and due dates are presented below:

1. **Team members (By October 2nd, 2020):** Each team must submit on Moodle a word file with the student names, ID number, and Concordia email of the members of the team.

2. **Preliminary report (By October 14th, 2020):** Each team must prepare and submit, through Moodle, a preliminary report of no more than three pages. This report must contain:
    a. a very short description of the tools the team is planning to use for the development of the project.
    b. a short description of the main entities/components that the team shall implement, i.e., input, real-time tasks, and output (e.g., GUI, terminal, etc.), as well as the approach the team shall follow.
    c. a chronogram with the envisioned main milestones and tentative dates.
    d. a very short description of the tasks to be performed by each member of the team.
3. **Final report (By December 7th, 2020):** Each team must submit a final report on Moodle. The guidelines to prepare this report will be presented in a separate document that will be posted on Moodle.
4. **Code (By December 7th, 2020):** Each team must submit on Moodle a ZIP file containing the implemented code, as well as a README.txt file with the basic instructions to run it.