Exercises 2IN60.9

Today you will write a controller application for controlling a real car via the CAN bus. You will need to work together with another group. It is up to you to divide the work between your group members.

You will test your application on a real car. However, each group will get to run their code only once, so make sure that your code is working!

9.1 Controlling a car via the CAN bus

In this exercise you will create a simple controller application. In particular, you will implement a task which listens to a message on the CAN bus indicating that the footbrake was pressed, and in response send a message which will actuate the engine speed meter (or rev counter) on the dashboard. The CodeWarrior project for this exercise is in the directory `exercise9_1`.

1. Study the task specification of `Task3` and implement the task.

   Hint: The payload of a CAN message contains up to 8 bytes of data. Consequently, the data may contain values for several devices, e.g engine speed, brake pressure for individual wheels, ... The format of the payload of each message is defined for every car. You are therefore given two utility methods for properly formatting the message payload for the Lupo. Use the `ExtractFootbrakeFromData()` function to extract the footbrake value form the incoming message, and the `InsertEngineSpeedIntoData` function to construct the outgoing data containing the engine speed value at the right place in the message payload.

   Hint: It turns out that Lupo continuously sends messages with the `EngineSpeedId`. These messages also contain other data than the engine speed. In order to avoid sending bogus vales for those other data with our engine speed, `Task4` captures the messages with the `EngineSpeedId` which are sent over CAN and stores the most recent one in the `SpeedData` global variable. For sending our desired engine speed value to the dashboard, you should change only the engine speed value in the data stored inside `SpeedData` and send that over CAN.

   Make sure to review your code within your group. If you don't feel confident, you may also implement a "simulated car" on a second board and test the communication with it. Use your time wisely!

2. Upload your code to your board and label the board with the provided label, so that during the testing you can find the board with your code and connect it to the car. The programmers will not be available during the testing!