**Bus-off simulation**

Objective: simulate a bus-off attack in a CAN bus with 2 ECUs connected to the bus. The two ECUs are V (victim) and A (attacker).

1) using a real log file build CAN frames sent by V. Pay attention in building frames in a way periodic messages have always the same arbitration ID.

2) simulate n runs (writing the code that allows V to continuously send ad-hoc CAN frames on the bus) and write the code that allows A to observe any pattern in V's transmissions (maybe and idea is to run V’s transmissions n times in a way A has enough data to identify a pattern, so the desired message M2 and its preceded ID).

3) once the pattern is identified, write the code that allows A to fabricate a new frame M2' with the same ID of real frame and that differs from it for 1 dominant bit in the control bits. This will allow M2' to win the arbitration with authentic M2.

4) write the code that allows A to listen the bus until the preceded ID of M2 is detected (so the ID of message M1) and, once that happens, A waits for the end of M1 transmission (3 bit-time after the M1's EoF - 10ms) before injecting M2'.

5) once M2' is transmitted, SocketCAN library will simulate for us the rest of busoff attack continuously retransmitting M2 and M2' until V is forced to error passive state and then to busoff. If socketCAN doesn't allow to do that consider only the next point 5).

5) once M2' is transmitted at the same time of M2, the collision will cause an error which increments both TECs in a way V firstly reaches error passive state and then busoff. Simulate this behaviour implementing both messages automatic retransmission, the different flags emitted by V and both TEC and REC counters for V and A.