# EOOS RT Automotive Demo Project for Versatile Application Baseboard

EOOS RT is an embedded object-oriented real-time operating system (RTOS) complied with MISRA C++ rules. It has been written in C++ language (ISO/IEC 14882:1998) and aimed to be used into microprocessor-based systems.

This project aims to cover microcontrollers which usually have limited hardware resources by AUTOSAR Adaptive Platform and to have possibility to execute AUTOSAR Adaptive Application on such kind of microcontrollers.

## System requirements

The project is being developed using Ubuntu 18.04 and all stuff is tested on this version of operating system.

## How to obtain environment

All necessary programs which are needed for building, demonstrating, developing and debugging can be obtained for the naked OS by executing 'get-environment.sh' script. If OS already has some programs installed on it, it is better to execute commands from the script manually in a console.

```
REPOSITORY/scripts$ source get-environment.sh
```

## How to build the project

The project can be built by executing 'build.sh' script.

```
REPOSITORY/scripts$ source build.sh
```

## How to run the project

Having built the project, it can be run on QEMU emulator by executing 'run-qemu.sh' script.

```
REPOSITORY/scripts$ source run-qemu.sh
```

## How do debug the project

The built project program can be debugged in a variety of ways a developer knows. Here we will introduce only two ways how we do this. The first way is to use terminal GDB debugging, and the second one is to use Eclipse IDE, which is of course more convenient way.

For both of the ways, in separate terminal the next command has to be executed.

```
REPOSITORY/scripts$ source run-qemu.sh --gdb
```

This command starts QUEM ARM emulator, loads the program, and freezes it  for waiting a connection of GDB client.

## Terminal GDB debugging

To connect to GDB server and the frozen program, the next command has to be executed in other terminal.

```
REPOSITORY/scripts$ source run-gdb.sh
```

This command starts GDB client and allows to execute any GDB commands in the terminal. For instance, the next command list demonstrates how to connect to the GDB server running on QUEMU and go through the program.

```
(gdb) target remote localhost:1234
(gdb) b main
(gdb) continue
(gdb) b _terminate
(gdb) continue
(gdb) quit
```

## Eclipse IDE debugging