# D2.6

# Micro-ROS default simulation environment release
# Revised

| | |
|---|---|
| Grant agreement no. | 780785 |
| Project acronym | OFERA (micro-ROS) |
| Project full title | Open Framework for Embedded Robot Applications |
| Deliverable number | D2.6 |
| Deliverable name | Micro-ROS default simulation environment release |
| Date | December 2019 |
| Dissemination level | public |
| Workpackage and task | 2.2 |
| Author | Juan Flores Muñoz (eProsima) |
| Contributors | Borja Outerelo Gamarra(eProsima) |
| Keywords | Hardware-Bridge, requirements, ROS, ROS2 |
| Abstract | |

# Contents

# 1 Summary

In this document, we will continue the study performed on the previous deliverable **D2.5 micro-ROS default simulation environment release**. Micro-ROS simulator is a required tool, which aims to simulate micro-ROS full-stack(Or almost the majority of the layers) on a regular PC. A simulator can be a useful tool because it can be used to try micro-ROS without specific hardware, also allowing the development of a continues integration tool for micro-ROS. In this document, we will start with a little brief of the achieved points on the previous deliverable, we will continue analysing the improvements performed by the community on the selected tools, and we will finalise with a conclusion of the state of this tool after a development period.

# 2 Acronyms and keywords

| Acronym | Definition |
|---|---|
| 6LowPAN | IPv6 over Low power Wireless Personal Area Networks |
| YAML | YAML Ain't Markup Language |
| SPI | Serial Peripheral Interface |
| ROS | Robot Operating System |
| UART | Universal Asynchronous Receiver-Transmitter |
| RTC | Real-Time Clock |

# 3 Link to software repositories

## 3.1 NuttX Official Repository

- Git repository: https://bitbucket.org/nuttx/nuttx
  - Branch: master
  - Commit: c86fabb

## 3.2 micro-ROS-simulator

- Git repository: https://github.com/micro-ROS/micro-ROS-simulator
  - Branch: stm32
  - Commit: 7fa6521

## 3.3 Qemu_stm32

- Git repository: https://github.com/beckus/qemu_stm32
  - Branch: stm32
  - Commit: 96ba2f1

# 4 Introduction

Micro-ROS simulator is a tool that allows us to run micro-ROS on a regular PC, giving the possibility to establish communication with any other micro-ROS device available on the network. This tool can give us some exciting possibilities if we achieve an excellent hardware simulation, like for example: - Try micro-ROS without the required hardware. - A possible solution for a future continuous integration tool, avoiding the usage of real hardware and related problems.

We have started from the work performed on the deliverable D2.5, in which we developed a study of which tools are available for our purposes and if it fits our requirements. After this research, we elaborate the next list of possible candidates:

- Official NuttX simulator + micro-ROS.
- Eclipse GNU QEMU.
- QEMU STM32.

To elaborate on the previous list, we have followed the next criteriums, base on the micro-ROS Client requirements observed on the real hardware.

- Emulate Nuttx or being able to execute NuttX.
- Support for network communications or/and serial communications with the host PC.
- C++ support.
- Timer and scheduling support.

After test every proposed tool and elaborate an in-depth study (Available on D2.5), we concluded that QEMU STM32 was the most suitable option for the simulator requirements. This tool can execute NuttX + micro-ROS client achieving communication with a micro-ROS Agent executed on the host PC. QEMU is an architecture emulator, and our development was base on the work previously performed by Beckus STM32. The original Qemu repository gives support for a generic Corte-M3 architecture, but this fork goes beyond the original and implements the STM32F103 MCU (Cortex-M3) with UART, timer and RTC support. This specific MCU is a similar model to the default selected for this project so that we could expect similar behaviour.

After performing an in-depth study of the internal structure and a later optimisation to obtain the best performance, we achieve a functional simulation. Still, unfortunately, the behaviour was very inconsistent with the random crash situation. Later investigation performed, obtains that probably the cause of this instability was the early stage of development of the Cortex-M3 architecture simulation, and being specific, the memory management is not adequately simulated.

After this conclusion, we considered created from scratch a simulator. But finally, we discarded this idea due to this could be an independent project by itself, that could take considerable time and effort. So, we focus on try to improve the stability of the tool.

# 5 Simulation improvements

Although our simulation choice was QEMU STM32, we continued following the improvement of the NuttX simulator, which is a pretty exciting tool because we avoid the hardware emulation simplifying

the tool. So, on the next points, we are going to expose the improvements developed by the community on each tool.

## 5.1 NuttX Simulator Status

During the last month, the NuttX community they add some interesting improvements, that we can see on the simplified list:

- Improve networking tools to make internet connection easy. 1,2
- Improvement on the fake interruption simulation. 3
- Improvement on the simulated timers. 4

The improvements have been varied, and they are focused on giving a more realistic experience, but our main problem was the lack of support for the C++ library(ulibc library), and this problem persists. Unfortunately, at the moment this tool was discarded due to the lack of C++ support, which will make impossible to run micro-ROS applications using that API.

## 5.2 QEMU STM32

This simulator has received the next improvement during the last months:

- New versions of Glusters libgfapi.so. It is a useful feature to work with file servers.
- Some minor fix on code.

As we can see, there isn't any real improvement, so we can consider that we have the same previous version and the development is freeze.

## 5.3 Configuration optimization

As we said previously, the main problem of QEMU STM32 is its inefficient memory management. Memory management impacts directly on the stability of the system, which causes a random crash on a high memory usage situation.

So, one of the solutions that we performed to improve this situation was to optimise micro-ROS usage. We made the next improvements:

- Disable every useless characteristic of NuttX and leaving as light as possible.
- Reduce the memory usage of micro-ROS by reducing the MTU and message Queue size.
- Disabling all the peripherals except the necessary such as UART, timer, and RTC.

All the previous changes return a slight stability improvement, but still an unacceptable number of random crash situations.

# 6 Conclusion

After this research, we think that continue the development of this tool it is not worthy as per the status at the moment of this deliverable. From one side, the NuttX simulator is getting some exciting improvement, but after some talks with the creator of the RTOS, they are not planning to give C++ support soon. From the other side, QEMU STM32 looks very promising, because we were able to run micro-ROS full stack and achieve external communication. But the performance is weak, and it seems that development has been abandoned because they don't commit anything relevant since May of 2019.

So, at this moment, there aren't any valid options that could fit our requirements.