D7.13

Long-term maintenance and evaluation plan
Initial Y1

Grant agreement no.	780785
Project acronym	OFERA
Project full title	Open Framework for Embedded Robot Applications
Deliverable number	D7.13
Deliverable name	Long-term maintenance and evaluation plan
Date	June 2019
Dissemination level	public
Workpackage and task	7.4
Author	Iñigo Muguruza Goenaga (Acutronic Robotics) and Ingo Lütkebohle (BOSCH)
Contributors	Borja Outerelo Gamarra (eProsima)
Keywords	micro-ROS, robotics, ROS, microcontrollers
Abstract	Preliminary plan for long-term maintenance and evolution of micro-ROS after the end of the project.

Contents

1 Acronyms and keywords

Term	Definition
ROS	Robot Operating System
OSRF	Open Source Robotics Foundation
MCU	Microcontroller Unit
CPU	Central Processing Unit
RTOS	Real-time Operating System
DDS	Data Distribution Service
XRCE-DDS	Extremely Resource Constrained Environments DDS
CI	Continuous Integration
RCL	ROS Client Library
RMW	ROS Middleware Interface
RCLC	ROS Client Library for C language
RCLCPP	ROS Client Library for Cpp language
TSC	Technical Steering Committee
LTS	Long Term Support
WG	Working Group
SIG	Special Interest Group
AL	Abstraction Layer

2 Intro

In this document, we will introduce the OFERA long-term maintenance and evolution plan of the relevant project results. To provide context, the OFERA project aims to bring resource-constrained devices, such as microcontrollers, as first-class participants in the robot ecosystem. In particular, it aims to offer support to the Robot Operating System (ROS) on its second version: ROS 2. The document will introduce the current strategy towards making the results accepted, maintained and further developed.

3 Structure

First of all, we provide background information about ROS, where we describe the development status, how the software is organized and the new governance structures that have been created by the Open Source Robotics Foundation (OSRF). Secondly, micro-ROS software entities possible accommodation in ROS 2 source-code is depicted, where the similarities and the divergences are explained, package by package. We also present the candidates which we want to be in charge of for each package. Thirdly, the conclusion is presented.

4 Background on the ROS2 development model

This projects target ROS2, which already integrates basic requirements for real-time, industrial and embedded-system use (see why ROS2) and thus provide a sufficiently mature base.

ROS2, like ROS before it, is organized into modules called "packages", which are the basic unit of dependency management. Active packages are looked after by a developer, called "maintainer". This maintainer could be a member of the Open Robotics, a company developer or an individual that overviews, releases and manages the package. ROS provides a maintenance guide about the role the maintainer must perform.

Most importantly for the purposes of this document, when code is *contributed* to a package, the maintainer is responsible for checking it and, if accepted, is *thereafter expected to maintain it*. This transfers the maintenance burden and is therefore an attractive strategy to minimize future work, but of course, this approach has practical limits. When making large contributions, their continued maintenance cannot always be off-loaded to the existing maintainer, but the contributor is expected to shoulder some of this burden.

Temporally, the development and maintenance of ROS is organized into so-called distributions, where "a distribution is a versioned set of ROS packages". Within a distribution, API changes to the core are avoided, but when changing distributions, such changes may occur. Such API changes are currently more significant in ROS2, as it is in its early stages, feature and performance discussions are taking place.

Distributions are normally released every 6 months, and in principle, once a distribution is released, older distributions need no longer be maintained, except for so-called Long-Term-Support (LTS) distributions, which have a lifetime of (currently) 2 years. In practice, packages should also be maintained on non-LTS distributions for a grace period to allow switch-over. This period has traditionally been anywhere from a few weeks up to several months or even years. For ROS2, it has so far been common practice to drop support for older distributions immediately, but since the first LTS distribution ("Dashing Diademata") has just been released, this will now change.

Moreover, each distribution is typically released for several underlying OS platforms (currently Linux, OSX and Windows) and architectures (x86 and ARM, 32bit and 64bit).

Apart from bug fixes, the change from one distribution to another is the most significant source of maintenance effort. Moreover, changes to the underlying OS platforms during the lifetime of a distribution can also cause maintenance tasks, e.g. when a dependency has been updated.

4.1 ROS TSC and Work Groups

One of the novelties that ROS 2 brings is the creation of the TSC. The TSC is the committee that "is responsible for the technical direction the project takes" and is "made up of individuals who contribute materially to the project and/or individuals who represents organizations that contribute materially to the project" (taken from this document). In addition, Working Groups have been established, which aims to discuss and develop ROS 2 in specific areas, such as safety-critical, navigation, security or quality assurance. The consortium members have been involved in the creation and leadership of the Embedded System working group, which discusses the use of microcontrollers in ROS 2.

Aside the OFERA project, some consortium companies -Acutronic Robotics, BOSCH and eProsima- are members of the TSC. In this TSC, they contribute to ROS 2 development performing different tasks, sponsoring ROS 2 developers and contributing with discussions and implementation of new features in ROS 2 from BOSCH side; porting MoveIt! to ROS 2, setting build farms for ROS 2 benchmarks from Acutronic Robotics side or adapting DDS implementation for ROS 2 use from eProsima side. Furthermore, some companies -BOSCH and Acutronic Robotics- also are involved in the Real-time working group, where real-time specific topics are discussed. This shows the commitment the consortium members have regarding ROS 2.

Once we have overviewed the purpose of ROS 2 and its general characteristics, we will proceed to show the long-term maintenance plan for micro-ROS.

5 Overall Strategy

Coming back to the proposal, OFERA is a three-year project with the aim of enabling companies to create cheaper robot components using state-of-the-art technology. If there is not a long-maintenance strategy, the use of the created assets and its support could be discontinued, wasting the development efforts incurred.

Maintenance normally is a task that is overlooked by organizations in projects. It requires a proper planning and resource dedication, which normally implies a big effort. Same applies to the outcome OFERA project is developing. The consortium is already creating a considerable amount of software distributed among different repositories. These repositories contain core elements, such as middleware or client-library implementation, and other elements, such as benchmarking tools. The required expertise to maintain all this outcome is considerable.

Taking the explained context into account, the best chance to ensure that the effort the consortium is doing to develop micro-ROS, is to merge it into ROS 2 repositories. This will imply that the development effort would not be solely of the consortium companies, giving the chance to other companies and foundations to use, analyze or improve it. This scenario would allow having access to more resources that nowadays the project has, for example, in terms of man-power or the quality of the technical discussion happening, as there would be a larger amount of people taking care of the packages micro-ROS is composed of.

The relation of the source-code OFERA project is creating regarding ROS 2 is various, depending how the contributions fit in the actual code. We can split them into the following three kinds:

- Modifications to ROS2 to improve performance/resource use.
- Added features within the core libraries.
- Supporting software specifically for micro-controllers.

The next section analyzes each point in depth. But before, mention that this source-code's elements are scattered through all the micro-ROS layers, which are depicted in the next image:

5.1 Modifications to ROS 2

The aim of the modifications is to improve in overall ROS 2 performance. These modifications will come from different sources, for instance, the ROS 2 working groups or parties that are interested in

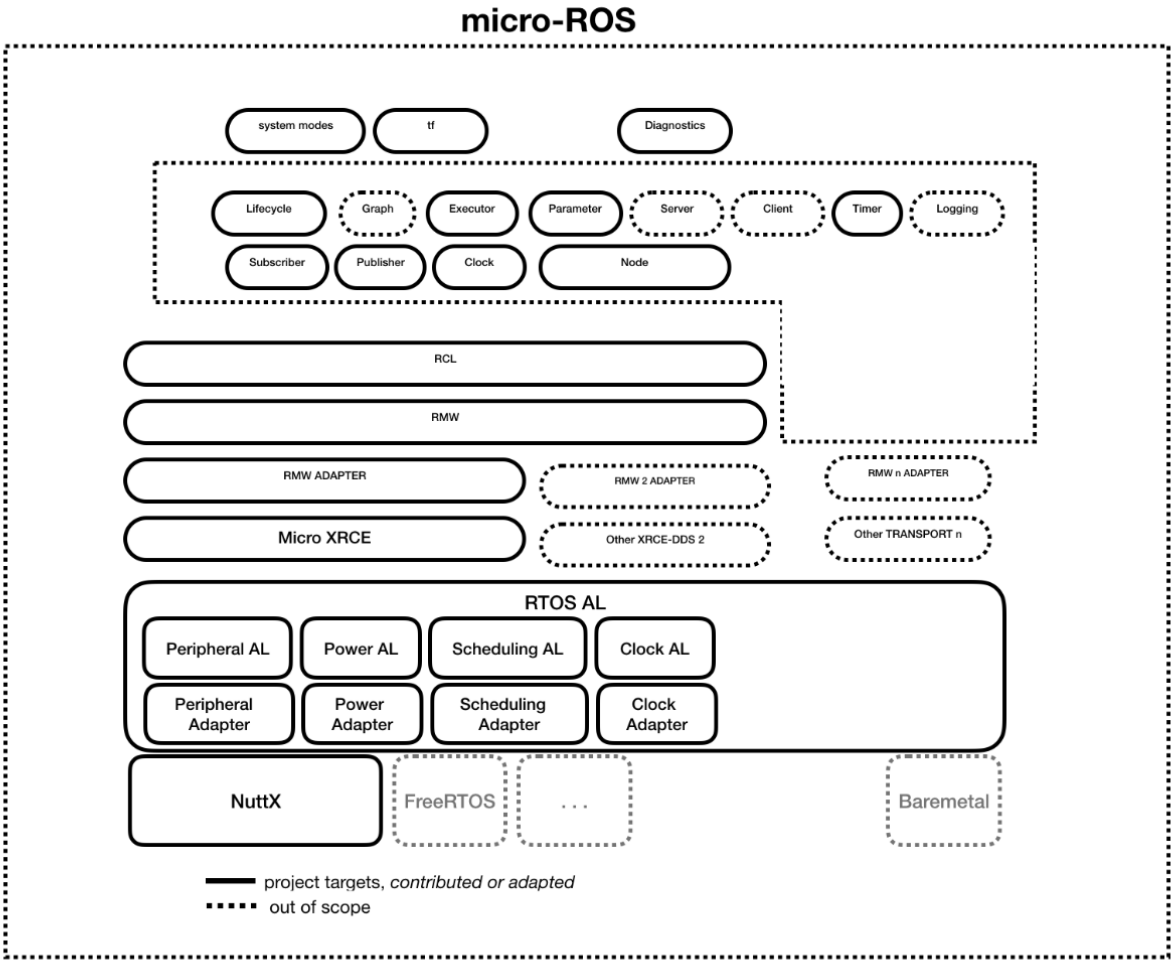

Figure 1: micro-ROS stack

improving a specific ROS 2 aspect. In our case, the consortium expects to rise technical discussion in fields that are related to microcontrollers, such as real-time determinism, memory footprint or memory allocation and fragmentation.

As ROS 2 has a layered stack, it is important to understand how each layer behaves and affects the overall performance. The use of test tools is the right method for gaining this comprehension. Thanks to the conclusions we can take from these tests, ROS 2 modifications and improvements could be proposed to the rest of ROS 2 developers.

The work package five gives the tools to the members for analyzing micro-ROS. In addition, consortium members also allocated some time to start analyzing tools for ROS 2 analysis. ROS 2 developers could use these tools for their daily tasks and, once the project is over, it is expected that the community should have adopted them, as it will be of their own interest to maintain and extend them.

Tool/asset/package	Software Layer	OFERA Maintainer	Long-term maintainer Candidate
Benchmarking tools	Other	PIAP	Open Robotics, Community
ROS 2 tracing instrumentation	Other	Bosch	Open Robotics, Community
ROS 2 trace analysis	Other	Bosch	Bosch

- **Bencharking tools**

The platforms that PIAP provide will allow to test the complete micro-ROS stack, starting from the RTOS itself, up to the application user wants to deploy in its robot. During the project period, this task is of PIAP's competence. But, as these tools are useful for the entire ROS 2 community, it is expected that the OSRF and/or the Embedded System work group to take over and further extend and maintain them. The benchmarking tools are intended for embedded software benchmarking with low/none overhead intrusion. These tools will be relatively easy to use and will allow simple analysis of the obtained data also by graphic visualizations.

- **ROS 2 tracing instrumentation and analysis**

The aim of this tool is to log the ROS 2 execution, to analyze it afterwards. Similar to earlier work for ROS1, the instrumentation itself belongs into the ROS2 core and OSRF has indicated that they will accept a pull request and maintain it afterwards.

Separate from that, a number of default analysis scripts will be provided by Bosch, and maintained by Bosch after the project. This has internally been agreed upon already for similar tools for ROS1.

5.2 Added Features within the Core Libraries

In the design process of ROS 2, many features that are wanted have been added to the development roadmap. These features are being developed gradually, as the effort is big and priorities of those

features varies. Part of the software packages that are being implemented in OFERA project, contributes to that list. In addition, its implementation takes in mind its use in micro-ROS, tackling both implementations.

The work package four of OFERA project is the one providing this kind of software entities, such as system-modes, diagnostics, embedded TF or executors.

As these software packages will supplement the ROS 2 core, it is expected that the maintenance will be performed by Open Robotics. If there is desire of extending those capabilities, the interested party should contribute and develop it.

Tool/asset/package	Software Layer	OFERA Maintainer	Long-term maintainer Candidate
Embedded TF	ROS Client library	BOSCH	Open Robotics
Real-time Executor	ROS Client library	BOSCH	Open Robotics
System modes	ROS Client library	BOSCH	Open Robotics, BOSCH
Diagnostics	ROS Client library	BOSCH	Open Robotics
Client library extensions	ROS Client library	eProsima	eProsima

- **Embedded TF**

Embedded TF provides various mechanisms to minimize the communication between the agent and a micro-ROS-based MCU. In addition it provides mechanisms to reduce the computational load on the MCU (e.g., by pushing filters to the agent). We expect that most of the mechanisms are of general interest for the community, independent of micro-ROS and can be integrated in the mainline TF code maintained by Open Robotics.

- **Real-time Executor**

The ROS 2 Executor concept is still under discussion, despite the first LTS release of ROS 2 in May 2019. In the same month, a Real-Time Working Group (WG) has been founded on initiative of Open Robotics and the ROS 2 TSC, with the goal to develop robust, real-time execution mechanisms for ROS 2. Several OFERA partners participate in this WG to ensure that the Executor concepts developed in micro-ROS are aligned with the concepts and implementations for micro-ROS and with the ultimate goal to bring them in the mainline code of rcl and rclcpp maintained by Open Robotics.

- **System modes**

The system modes package is developed based on the new ROS 2 lifecycle, which is part of rclcpp and maintained by Open Robotics. Open Robotics showed interest in the system modes concept and there are on-going discussions of bringing selected sub-concepts into the main packages.

- **Diagnostics**

The diagnostics package of ROS 1 is widely used. Although it belonging to the core ROS organization on GitHub, it is not maintained by Open Robotics but an external developer. In the OFERA project, Bosch started porting diagnostics to ROS 2 in early 2019, as an important dependency for further works on the system modes concept. We are already in discussion with Open Robotics and the ROS 2 TSC about a long-term maintenance by Open Robotics. A first step to be made soon is that the ported repository is stored in the ROS2 organization by Open Robotics on GitHub.

- **Client library extensions**

Even thought OFERA consortium has agreed to stick with the current existing ROS2 client libraries, RCL with modules and ROS2 RCLCPP, custom parts may be needed to be developed exposing functionalities required by the nature of the middleware used in the lower layers, mostly configurations. OFERA partner, eProsima, will extend existing libraries to expose the required APIs. Currently there is no a discussion regarding the long-term maintenance with Open Robotics, but the initial plan is to integrate these extensions as part of the RCL and RCLCPP in the upstream in ROS2 Organization from GitHub. Once we integrate our development with Open Robotics one, a discussion on the maintenance should take place.

5.3 Specific Software

As the aim of the OFERA project is to support ROS 2 in microcontrollers, there is code that only belongs to this resource-constrained devices. If we compare ROS 2 and micro-ROS stacks, we can see that the bottom part is where the most substantial differences are located. For example, micro-ROS makes use of a RTOS, which is not contemplated in ROS 2.

To minimize the support effort, it is of micro-ROS interest to keep this software entities that lives outside ROS 2 as few as possible. The following table shows which are those software entities and which is the expected long-term maintainer:

Tool/asset/package	Software Layer	OFERA Maintainer	Long-term maintainer Candidate
NuttX RTOS	RTOS	Acutronic Robotics	NuttX Community & Embedded system working group
RTOS ALs	RTOS	Acutronic Robotics	Embedded system working group or RTOS vendor

Tool/asset/package	Software Layer	OFERA Maintainer	Long-term maintainer Candidate
Micro XRCE-DDS	Middleware	eProsima	eProsima
`rmw` for Micro XRCE-DDS	ROS RMW	eProsima	eProsima
micro-ROS to FIWARE bridge	Other	eProsima	eProsima
micro-ROS to H-ROS bridge	Other	Acutronic Robotics	Acutronic Robotics
HRIM usage	Other	Acutronic Robotics	Acutronic Robotics

- **NuttX RTOS**

The project's RTOS is supported by Acutronic Robotics and it consist of a maintained fork of the original BitBucket NuttX repository. The fork ensures the support of the development platforms selected in the project. Nowadays, there is little divergence of both repositories, thus consortium members agreed on trying to get into upstream the changes performed in the fork, in order to be synced. This is going to be done sending patches of the development boards and drivers to the NuttX creator, Greg Nutt. This will allow benefiting of the NuttX community development effort and maintenance during the project and its end.

Even though NuttX RTOS would be supported by the community, the NuttX `colcon`-based compilation could require of maintenance, as this compilation tool belongs to ROS 2 and not to the RTOS. This is an ongoing work the consortium is performing, and, once finished, it is expected to be placed in a project's repository. This repository should be maintained by the Embedded System working group as it is the entity bridging between ROS 2 and the embedded devices.

While the project gains interest within the community and external companies, it is also expected to have support for other RTOSes, following the micro-ROS modular approach. We have already seeing interest of supporting FreeRTOS by community developers. In this scenario, it is likely to see development and maintenance effort of individuals or companies that are interested in having another supported RTOS within micro-ROS.

- **RTOS ALs**

The aim of the RTOS ALs is to provide resources to the upper micro-ROS layers. This includes clock, time and timer primitives, power management, scheduling interfaces or communication mean abstractions that are crucial for micro-ROS. This ALs also provides of APIs to abstract the RTOS that is beneath used, making micro-ROS layers effortlessly exchangeable.

Nowadays, the consortium has made preliminary probe-of-concept of these ALs and, still, performance tests are pending (for more information, consult deliverable 2.2). As discussed in that

deliverable, the ALs are possible to implement, but we need to determine their impact on the performance.

In this situation, once we determine if the ALs finally get into the default micro-ROS layers, they will require of maintenance. As they are still a concept, they will be discussed and jointly developed within the Embdedded WG and maintained by it afterwards.

- **Micro XRCE-DDS**

eProsima Micro XRCE-DDS is an implementation of the OMG DDS-XRCE standard. This middleware implementation enables the microcontrollers to use DDS, thus enabling micro-ROS to interoperate with ROS2 natively.

The nature of DDS does not make it usable in resource-constrained environments like microcontrollers, due to computation and memory restrictions. Addressing this issue, the OMG designed a new communication standard called DDS-XRCE. This standard allows small devices to communicate with DDS. The central architectural concept is the existence of tiny clients communicating with a relatively more prominent Agent, which have DDS capabilities. In such architecture, the Agent acts in DDS world on behalf of the clients.

eProsima is the developer of a first and open source implementation, called Micro XRCE-DDS. This implementation, apart of being used as part of micro-ROS layers, it is used in other different projects not related to micro-ROS, making it a valuable product for eProsima.

eProsima treats Micro XRCE-DDS as any other of their products providing customer support and continuously maintaining and improving it, both with the community and with internal developments.

- **`rmw` for Micro XRCE-DDS**

Mimicking ROS2 architecture, an abstraction layer over the middleware is present in micro-ROS. This layer, RMW, is in charge of abstracting upper layers from the middleware implementation underneath even from the protocol used, DDS in ROS2, DDS-XRCE in micro-ROS. eProsima developed a specific implementation of this layer for its DDS-XRCE implementation, Micro XRCE-DDS. eProsima developed this layer the same way Fast RTPS implementation layer is in ROS2. eProsima will integrate this as a regular ROS2 package and keep it up to date related to:

1) Future Micro XRCE-DDS revisions due to implementation or DDS-XRCE standard changes.
2) RMW abstraction revisions from Open Robotics changes.

Currently, the rmw_microxrcedds package is hosted in a micro-ROS repository but, like the rest of micro-ROS packages will be proposed to be integrated withing ROS2 repositories.

- **micro-ROS to FIWARE bridge**

eProsima is the owner of FIROS2 FIWARE generic enabler, a component to bridge between several middleware components of FIWARE, including the FIWARE Context Broker and Fast RTPS, with ROS 2. As part of the OFERA project, eProsima will extend this bridge to incorporate micro-ROS support. Once this extension is done, eProsima product will be able to bridge micro-ROS devices and FIWARE components. This product, like any other product from eProsima, will be developed and maintained by eProsima.

- **micro-ROS to H-ROS bridge**

This bridge connects micro-ROS technology to H-ROS. As a result, micro-ROS enabled devices will be able to join robots that are based on H-ROS. As H-ROS is owned by Acutronic Robotics, they will support this bridge also in future.

- **HRIM**

Same applies to HRIM. HRIM is an open information model used under H-ROS robot bus. The support of HRIM for micro-ROS will be done by the owner Acutronic Robotics.

6 Conclusion

To sum up, we can see that the feasibility of introducing the developed software in OFERA project into ROS 2 is different, depending on the nature of each software entity. Taking this into account, we have set goals to try to merge them into ROS 2 development and maintenance mechanisms, following different strategy for each group. As the project is still ongoing, there is still work to be done and this first approach will be reviewed in the next deliverable, that is going to be published in month 36 of the project.