2.3

micro-ROS default RTOS release Software Release Y1

Grant agreement no.	780785
Project acronym	OFERA (micro-ROS)
Project full title	Open Framework for Embedded Robot Applications
Deliverable number	2.3
Deliverable name	micro-ROS default RTOS release – Software Release Y1
Date	December 2018
Dissemination level	public
Workpackage and task	4.2
Author	Acutronic Robotics
Contributors	
Keywords	micro-ROS, robotics, ROS, microcontrollers, RTOS
Abstract	This document explains the how the first release of the RTOS has been done, which test procedures have been followed. It also provides links to the released software and related documentation.

Contents

1 Summary

The first micro-ROS release offers a RTOS tested for the Reference Hardware Development Platforms (RHDP), (STM32L1 Discovery and Olimex STM32-E407 boards), selected in the deliverable *2.1 Report on the reference hardware development platforms*.

The microcontrollers used under these RHDPs could be considered medium to big size due to their technical characteristics in terms of the MCU they contain - an ARM Cortex-M3 and Cortex-M4F-, RAM and flash storage. As explained in the deliverable D2.1, the hardware resources they contain makes these microcontrollers capable, in principle, of executing the complete micro-ROS stack.

Even that we have focused on the RHDPs in this release, NuttX, by default, supports many development boards and microcontrollers. This implies that running the same applications in the rest of supported boards by the RTOS should be fairly easy.

Since the start of the project, many testing procedures have been made. This tests allowed us validating the hardware functionality the RTOS needs to provide for executing the project's software stack. These tests comprises of examples which uses communication buses and protocols such as UART, I2C or SPI, as well as UDP and TCP based networked communications.

This work has generated as outcome a first release where basic tests are also at user disposal. These tests are based on NuttX `defconfigs` created for each RHDPs, where developer can use them to compile the RTOS together with the test application, allowing the user to get easily familiar with the RTOS.

In addition to these `defconfigs`, tutorials have been developed and placed in GitHub's official documentation page, and under project's NuttX repository issues, to make the learning curve smoother.

In addition to the work performed under NuttX, we also have added the support for micro-XRCE-DDS client implementation and a micro-ROS client example. These allows users to test the complete stack which micro-ROS is comprised of. For that purpose, Docker files that allows to flash example binaries into target boards are provided in the micro-ROS docker repository.

2 Acronyms and keywords

Term	Definition
RTOS	Real-Time Operating System
AL	Abstraction Layer
XRCE-DDS	Extremely Resource Constrained Environments Data Distribution Service
UART	Universal Asynchronous Receiver Transmitter
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
CAN	Control Area Network
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
RAM	Random Access Memory

Term	Definition
RHDP	Reference Hardware Development Platform
AL	Abstraction Layer
GPIO	General Purpose Input Output

3 Overview to Results

This document provides links to the released software and documentation for deliverable D2.3 *micro-ROS default RTOS release Initial* of the Task 2.2 *Platform lower level firmware and libraries*.

As an entry-point to all software and documentation, we created a dedicated webpage on the micro-ROS website: https://microros.github.io/rtos/

The annex includes a copy of this webpage and a copy of the major documentation file of this software release.

4 Links to Software Repositories

NuttX repository, based on NuttX 7.26 release:

- Git repository: https://github.com/microROS/NuttX
  Version: NuttX 7.26
  Branch name: master
  Squashed commit: f93cc44

NuttX Apps repository, based on NuttX 7.26 apps release:

- Git repository: https://github.com/microROS/apps
  Version: NuttX 7.26
  Branch name: master
  Squashed commit: 700cdac

Markdown source of the microros.github.io/rtos webpage:

- Git repository: https://github.com/microROS/microros.github.io
  Squashed commit: c51748a

5 Annex 1: Webpage on RTOS

Content of the webpage at December 2018th, commit be163b3.

5.1 RTOS

5.1.1 Introduction

The use of Real-Time Operating Systems (RTOS) is a general practice in nowadays embedded systems. These embedded devices typically consist of a resource-constrained microcontroller that executes an application where the interaction with external components is performed. In many cases, this application contains a time-critical task where a time-deadline or deterministic response is required.

Bare-metal applications are also used nowadays, but it requires a very low-level programming skills and lacks of hardware abstraction layers that RTOSes offers. On the other hand, RTOSes typically uses hardware abstraction layers (HAL) that eases the use of hardware resources, such us timers and communication buses, making easier the development and allowing the reuse of code. In addition, they offer thread and tasks entities that, together with the use of schedulers, provides the necessary tools to implement determinism in the applications. The scheduling normally consists of different algorithms where the user can choose from. Another feature that RTOSes normally offers is the stack management, helping in the correct memory usage of the MCU, a valuable resource in embedded-systems.

5.1.2 RTOS in micro-ROS

Due to the benefits explained in the introduction, micro-ROS integrates RTOS in its software stack. The use of such a tool enhances the micro-ROS features and allows reusing all the tools and implementations they provide. As the micro-ROS software stack is modular, the exchange of software entities is expected and desired. Same happens with the RTOS. Even that NuttX is the *default* RTOS for the project, it is expected that several of them could replace it.

5.1.3 NuttX RTOS

NuttX is a RTOS that emphasizes its compliance with standards (such as POSIX) and small footprint, it can be fit in 8 to 32 bit microcontrollers. The use of POSIX and ANSI standards, together with the mimic it does to UNIX APIs, makes it friendly to the developers that are used to Linux. The RTOS is licensed under BSD license and makes use of GNU toolchain. In order to obtain more information, please visit NuttX overview page.

5.1.4 Supported development boards

For development purposes, project consortium has established two development boards as development blueprints: Olimex STM32-E407 and the STM32LDiscovery. Both development board schematics are open, promoting new hardware designs based on such a platforms.

The first one consists on a ARM Cortex-M4F MCU with 196 KB of RAM and 1 MB of flash. It also offers Arduino-like expansion pins and Ethernet communication means.

The STM32LDiscovery board contains a STM32L ultra-low power packaging and consists of a ARM Cortex-M3 MCU that integrates 32KB of RAM, 256KB of flash memory. This microcontroller aims to target low-power applications.

5.1.5 Getting started with NuttX and micro-ROS

In order to obtain more information about how to get started using this RTOS, please check our documentation repository, where tutorials and getting started material is offered.

We have created several Docker containers, where some are meant for development purposes and other to execute precompiled examples. These Docker files have been gathered together in the micro-ROS Docker repository.

6 Introduction

The default OFERA project's first RTOS release, is based on official NuttX 7.26 release. This release has been tested and updated for the needs of OFERA project.

Following the selection RHPDs made at deliverable number 2.1, *Report on the reference hardware development platforms*, different RTOS features have been tested under the two selected development boards. This includes testing different communication buses using sensors, testing wired and wireless communications as well as testing additional features, such as the NuttX Power Management.

At the beginning of the project, a fork of that moment's release was done, based on version 7.23. Using this version, the validation and selection of the RHDPs was made. Once this task was performed, it was considered that having the fork updated was interesting. This is due to the fact that some bugs and new features where placed in the subsequent releases of the RTOS at the official NuttX repository.

In order to do so, we have added commits to the NuttX 7.26 release, to add support for the development boards and to offer some basic test example where consortium members and developers could make use of.

NuttX RTOS release comprises of two main repositories which are of public domain: the `NuttX` repository, where the RTOS skeleton is placed and the `apps` repository, where applications and examples are stored.

Apart from this repositories, basic Docker files are provided, that are updated to this release, in order to ease and speed-up the setting of the tools for NuttX compilation and tests. These Docker files, clones the compilation tools, configures the tool-chain depending the target and downloads

the micro-ROS NuttX repositories. Thanks to this, user can just start testing code under its own development board.

In addition, this first release also supports Micro XRCE-DDS client implementation example, provided by eProsima. Micro XRCE-DDS is a XRCE-DDS implementation that micro-ROS stack uses. Examples and tutorials have been also uploaded, together with a Docker container, so developers can easily test it. We also have added an example of micro-ROS client which is comprised by the complete stack. This example uses NuttX RTOS, Micro XRCE-DDS DDS implementation and micro-ROS ROS 2 user library adapted to microcontrollers.

The abstraction layers provided for abstracting RTOSes is not provided by together with the RTOS. This is because these files does not contribute meaningfully to the improvement of the RTOS itself. Additionally, it is beneficial to maintain both tools in a separate repository, so the development work of both tools are not mixed. For more information about abstraction layers, please read the deliverable number 2.2.

7 Supported features

This section aims to explain the main RTOS features which has been tested under the RHDPs.

7.1 Peripherals

Peripherals play a key role in deeply-embedded systems, where interfaces are required to communicate with sensor and actuators. In first place, the most typical communication protocols were tested. Serial communication was mentioned the project proposal, for having a baseline for communicating with external devices, such as XRCE-DDS Agent as well as serial-based sensors. I2C and SPI are also widely used in inexpensive and/or small sensors, this is why they have been also tested.

Peripherals has been tested using sensor test applications or using communication test applications, in the last ones, networking tests were performed.

7.1.1 UART

Serial communication has been tested in both boards at first instance. It has been tested using NuttX command line interface tools called `NSH`. This is taken as the most basic validation step in a NuttX compilation for development boards.

7.1.2 I2C

NuttX I2C master controller has been tested using `HIH6130` and `BMP180` sensors. These sensors have been attached to the external GPIOs the development board exposes.

Using example applications stored under `app` repository and using custom `defconfigs`, both RHPD's I2C functionality validation was made.

7.1.3 SPI

SPI controller has been tested in both RHPDs platforms using `MRF24J40` wireless chip as base, verifying it works.

7.2 Networking

Networking capabilities also plays a important role under the project. This is because the middleware requires the support for serial, TCP and UDP communication protocol layer support for connecting clients to the agent. As the STM32L1 Discovery lacks of Ethernet interface, the networking tests has been made only in the Olimex STM32-E407. Additionally, NuttX 6LOWPAN stack has been tested. Notice that, 6LOWPAN stack doesn't fit in the STM32L1, due to its RAM size.

7.2.1 TCP

NuttX contains a TCP/IP network stack implementation. A TCP echo example has been tested, where the development board was holding a server where, re-sends all the received packages.

7.2.2 UDP

Same as TCP, UDP communications where tested using a UDP echo server. The development board has been plugged into a local network, where an external computer was sending UDP datagrams to the server. Server was resending the packages back to the client.

7.2.3 6LOWPAN

Executed 6LOWPAN example uses two Olimex STM32-E407 boards where one acts as UDP server and the other as client, where wireless interfaces are set up using NuttX `i8` tools. Notice only that point to point communication type is used, not using mesh capabilities 6LOWPAN offers.

7.3 Additional features

NuttX offers some additional features that could be included in the build time of the RTOS. These features are helpful for system diagnosis and debug, as well as for controlling the power consumption of the devices at run time, among many other features.

7.3.1 NSH

The NuttX Shell is a basic shell application that could be integrated at compilation time for basic introspection of the target embedded system. It always has been included at the tutorials at both boards, due to its usefulness.

7.3.1.1 NSH over UART

NSH could be configured to use a serial interface. This application has been included at both development boards example `defconfig`s.

7.3.1.2 NSH over USB

NSH also could be configured to use an USB port for communication purposes. As only the Olimex STM32-E407 has this port, this development board has been used to validate that the RTOS has this feature working.

7.3.2 Power Management

Power management is an relevant feature within OFERA project, it allows to deeply embedded systems, to consume less power and lasting more when batteries are used for feeding the electronics. That is why the NuttX power management tool has been also tested.

7.3.3 CPU load and RAM usage

Similar to a microprocessor based operating system (OS), NuttX offers NSH command-line tools in order to know the CPU load and the available RAM at the MCU. We have also used this tool for measuring the memory usage of each example application.

8 Application examples

In order to facilitate getting started with NuttX, we have created NuttX `defconfig` files for both development platforms, as well as we have placed a tutorial document explaining how to use them. As external hardware is also used in the examples, we have also added pictures showing how to connect them to the development board's expansion pins. The tutorials can be found at here, it is a living document that will be updated as new features are tested.

8.1 Defconfig files for Olimex STM32-E407

The defconfig files for this board can be found here. This is the status of the `defconfig` files provided at the NuttX repository at the time of this release:

Minor updates to be done at the table

Defconfig	Status
ADC	Tested
BMP180	Tested
HIH6130	Tested
MRF24J40-6lowpan	Tested
MRF24J40-mac	Tested
netnsh	Tested
nsh	Tested

Defconfig	Status
PM	Tested
SD	Tested
TCP Echo	Tested
Telemetry	Tested
Telnet	Tested
Timer	Tested
UPD echo	Tested
Web Server	Tested
micro-XRCE-DDS	Tested
micro-ROS	Tested

8.2 Defconfig files for STM32L1 Discovery

The defconfig files for this board can be found here. This is the status of the `defconfig` files provided at the NuttX repository:

Defconfig	Status
BMP180	Tested
HIH6130	Tested
MRF24J40-mac	Not Tested
nsh	Tested
PM	Tested
micro-XRCE-DDS	Tested
micro-ROS	Pending, too big to its RAM

8.3 Micro XRCE-DDS Client test

A basic Micro XRCE-DDS client test has been also added to NuttX repository. In this example, the microcontroller runs NuttX with Micro XRCE-DDS client binary. It exchanges data with a Micro XRCE-DDS agent, using serial communication protocol.

8.4 micro-ROS Client test

A basic micro-ROS client test has been also provided by eProsima. In this example, the microcontroller runs NuttX with Micro XRCE-DDS client and ROS 2 client binary adapted for microcontrollers. It exchanges data with a Micro XRCE-DDS agent, using UDP protocol.

9 Conclusions

NuttX development is an ongoing process, where bug fixes and the addition of interesting features are regularly made. In order to benefit of this situation, ALR will continue monitoring the status of

the development of NuttX and merge them in the micro-ROS NuttX repository.