Towards ROS 2 micro-controller meta cross-compilation

ROSCON 2018, September 30th, Madrid

Iñigo Muguruza Goenaga, Juan Muñoz Flores, Víctor Mayoral Vilches (Erle Robotics)
Loïc Dauphin, Emmanuel Baccelli, Cedric Adjih (INRIA)
Discussed topics

- Previous work mention with µCs … and new approaches
- Meta cross-compilation approach
- Examples
- Call for collaboration
Previous Work

• **ROSSerial (ROS 1)**
  - Uses a proxy to integrate μC pub/subs in ROS network
  - Based on Arduino-like μCs,
  - Based on serial/USB based and limited to 32 bytes data

• **ros2_embedded_nuttx (ROS2)**
  - Required additional external RAM
  - Complex to port complete DDS (Tinq)

• **FreeRTPS (ROS2)** => baremetal, small RTPS implementation, μC in mind

• **uROSnodel (ROS1)**
Novel Work

New approaches have arisen, confirming the interest of community to support ROS 2.0 in µCs.

**OERA** project

- Complete ROS 2.0 stack based for µCs (XRCE-DDS oriented)
- ROSSerial for ROS 2.0*

*new approach being considered
Issues

Support for μC in ROS 2.0 difficult because:

- RAM !
- C++ 11/14 => Not supported for μC toolchains
- Toolchain are different: CMake vs Makefile
Brainstorming outcome

Approaches to support µCs:

1. Add support CMake to each RTOS/platform

2. Support RTOS-dependant compilation process in ROS 2.0

3. “meta cross-compile” => Convert ROS 2.0 code into RTOS-centric code
Meta Cross-Compilation

1. ROS2 package
   - package.xml
   - CMakeLists.txt
     - include
       - header1.h
       - header2.h
       - ...
     - src
       - source1.c
       - source2.c
       - ...

2. Compilation
   - Ament
   - Ament
   - Ament
   - Ament
   - ament2riot toolchain file
   - RIOT module
     - Makefile.include
     - Makefile
     - source1.c
     - ...

3. Microcontroller Binary file
   - RIOT
     - Makefile
     - system
     - Microcontroller
     - Binary file

4. NuttX app
   - Ament
   - Ament
   - ament2nuttx toolchain file
   - Nuttx
     - Makefile
     - source1.c
     - ...

5. Microcontroller Binary file
   - Nuttx
     - Makefile
     - system
     - Microcontroller
     - Binary file

6. Meta cross-compilation
   - Ament

Erle Robotics
Meta Cross-Compilation

https://github.com/erlerobot/riot-ros2/blob/nuttx/ament2nuttx.cmake
Meta Cross-Compilation

```cmake
# NuttX Kconfig composition

set(KCONFIG "${CMAKE_INSTALL_PREFIX}/${target}/Kconfig")
file(WRITE "${KCONFIG}" "config EXAMPLES_ROS\n")
file(APPEND "${KCONFIG}" "bool \"Hello, micro-ROS!\" example\"\n")
file(APPEND "${KCONFIG}" "default n\n")
file(APPEND "${KCONFIG}" "---help---\n")
file(APPEND "${KCONFIG}" "Enable the \"Hello, micro-ROS!\" example\n")
file(APPEND "${KCONFIG}" "if EXAMPLES_ROS\n")
file(APPEND "${KCONFIG}" "config EXAMPLES_ROS_PROGNAME\n")
file(APPEND "${KCONFIG}" "string \"Program name\"\n")
file(APPEND "${KCONFIG}" "default \"hello\"\n")
file(APPEND "${KCONFIG}" "depends on BUILD_KERNEL\n")
file(APPEND "${KCONFIG}" "---help---\n")
file(APPEND "${KCONFIG}" "This is the name of the program that will be use when the NSH ELF\n")
file(APPEND "${KCONFIG}" "program is installed.\n")
file(APPEND "${KCONFIG}" "config EXAMPLES_ROS_PRIORITY\n")
file(APPEND "${KCONFIG}" "int \"Hello task priority\"\n")
file(APPEND "${KCONFIG}" "default 100\n")
file(APPEND "${KCONFIG}" "config EXAMPLES_ROS_STACKSIZE\n")
file(APPEND "${KCONFIG}" "int \"Hello stack size\"\n")
file(APPEND "${KCONFIG}" "default 2048\n")
file(APPEND "${KCONFIG}" "endif\n")
```

https://github.com/erlerobot/riot-ros2/blob/nuttx/ament2nuttx.cmake
Proof of concepts

- ARM Cortex-M0+, 32KB RAM and 256KB Flash, Atmel ATSAMR21G18A
  - Using RCLC, RCL, RMW_NDN, NDN, RIOT => 78 KB Flash (~30%) and 10 KB RAM (~31%)

https://github.com/astralien3000/riot-ros2

- ARM Cortex-M4F, 256KB RAM and 1MB Flash, Atmel ATSAMR21G18A

https://github.com/erlerobot/riot-ros2/tree/nuttx

https://github.com/erlerobot/riot-ros2/tree/nuttx

Erle Robotics
Next steps & call for collaboration

• The interest of using ROS in µC is more alive than ever!

• Interested? Contact us! (inigo@erlerobotics.com)