micro-ROSの調査

動作確認バージョン

- ROS2 : Humble

作業環境

ビルド環境は下記を用意した。

1. Ubuntu PC : Ubuntu 22.04 64bitをインストールしたPC環境

   micro-ROSをビルドし、libmicroros.aや*.hをビルドする環境

2. Windows PC

   TOPPERS/ASP3やサンプルアプリをビルドする環境

3. Raspberry Pi : Raspberry Pi 3B+にUbuntu 22.04 64bitをインストールした環境

   エージェント（`Micro-XRCE-DDS-Agent`）を実行し、USBシリアルでターゲットボードと通信

Ubuntu PC の環境設定

micro-ROS開発環境の構築

micro-ROSをビルドしない場合は、11.の手順から行う。

0. 前準備 : ビルドを高速化するために以下を設定。NUMには並列コンパイルの数を記載する。

```
export MAKEFLAGS="-j NUM"
```

1. ロケールを設定するため、以下のコマンドを実行する。

```
sudo apt update && sudo apt -y install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
```

2. GPGキーを設定する。以下のコマンドを実行する

```
sudo apt update && sudo apt -y install curl gnupg2 lsb-release
sudo curl -sSL
https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg
```

3. リポジトリをソースリストに追加する。以下のコマンドを実行する

```
sudo echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/ros-archive-keyring.gpg]
http://packages.ros.org/ros2/ubuntu $(source /etc/os-release && echo
$UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list >
/dev/null
```

4. ROS2パッケージをインストールする。以下のコマンドを実行する

```
sudo apt update; sudo apt -y install ros-humble-ros-base
```

rviz2等をインストールする場合は以下もインストール.

```
sudo apt install ros-humble-desktop
```

```
source /opt/ros/humble/setup.bash
```

5. micro-ROSをインストールするため、以下のコマンドを実行する。

```
sudo apt update
sudo apt install -y python3-pip python3-nose clang-format pyflakes3 python3-
mypy python3-pytest-mock gperf ros-$ROS_DISTRO-osrf-testing-tools-cpp
python3-lttng ros-$ROS_DISTRO-mimick-vendor python3-babeltrace python3-
rosdep2 python3-colcon-common-extensions
export CROSS_COMPILE=/usr/bin/arm-none-eabi-
export TOOLCHAIN_PREFIX=/usr/bin/arm-none-eabi-
```

6. ビルドに必要な準備を行うため、以下のコマンドを実行する。

```
mkdir uros_ws
cd uros_ws
git clone -b $ROS_DISTRO https://github.com/micro-ROS/micro_ros_setup.git
src/micro_ros_setup
rosdep update && rosdep install --from-path src --ignore-src -y
colcon build
source install/local_setup.bash
cd ..
```

7. micro_ros_stm32cubemx_utilsをgitからクローンする。以下のコマンドを実行する。

```
git clone -b $ROS_DISTRO https://github.com/micro-
ROS/micro_ros_stm32cubemx_utils.git
```

8. micro-ROSファームウェアの構成を行うため、以下のコマンドを実行する。

   ARMのコンパイラをダウンロードするため時間を要する。

```
mkdir nucleo_f446re
cd nucleo_f446re
ros2 run micro_ros_setup create_firmware_ws.sh  freertos nucleo_f446re
```

   アプリはping_pongを選択する

```
ros2 run micro_ros_setup configure_firmware.sh ping_pong --transport serial
```

   ping_pongは、パブリッシャーが2つ`RMW_UXRCE_MAX_PUBLISHERS=2`、サブスクライバー2つ`RMW_UXRCE_MAX_SUBSCRIPTIONS=2`でビルドされるので、他のアプリより汎用的。

9. ビルドを実行する。

```
ros2 run micro_ros_setup build_firmware.sh
../micro_ros_stm32cubemx_utils/library_generation/toolchain.cmake
./micro_ros_stm32cubemx_utils/library_generation/colcon.meta
```

10. ヘッダーファイルとライブラリファイルを得る。

```
zip -t firmware firmware
```

    `firmware.zip`ファイルの中か下記のフォルダツリーとなっている。 TOPPERSアプリで必要なのは`mcu_ws`フォルダと`libmicroros.a`のみ。

```
@startuml

salt
{
{T
    firmware
    + dev_ws
    + freertos_apps
    ++ microros_nucleo_f446re_extensions
    +++build
    ++++libmicroros.a
```

```
    + mcu_ws
    + toolchain
}
}

@enduml
```

11. リポジトリをクローン

```
git clone https://github.com/exshonda/micro-ROS_ASP3.git .
```

下記のフォルダツリーになっている。

上記の手順でビルドしたlibmicroros.aを使用する場合は、firmware.zipのmcu_wsを下記のmcu_wsに展開する。 また、freertos_apps\microros_nucleo_f446re_extensions\build\libmicroros.aを、mcu_wsに保存する。

```
@startuml

salt
{
{T
    micro-ROS_ASP3
    + asp3
    + mcu_ws
    ++ libmicroros.a
    + mcu_ws_f767zi
    ++ libmicroros.a
    + micro_ros_asp
    + sample
    ++ joystick
    +++ deps
    +++ build
    +++ **Makefile**
    ++ libkernel
    +++ deps
    +++ build
    +++ **Makefile**
    ++ ping_pong
    +++ deps
    +++ build
    +++ **Makefile**
    ++ publisher
    +++ deps
    +++ build
    +++ **Makefile**
    ++ subscriber
    +++ deps
```

```
    +++ build
    +++ **Makefile**
    ++ svd
    ++ **Makefile.target**
    + README.md
    + **Makefile**
    + microros.code-workspace
}
}

@enduml
```

`mcu_ws_f767zi`フォルダは、上記8.の手順以降を`nucleo_f767zi`で行ったもの。

12. TOPPERS/ASP3とアプリをビルドする

    `sample\Makefile.target`で使用するターゲットボードの定義のみをコメントアウトする。

    上記で、`deps`と`build`フォルダがない場合は作成する。

```
make all
```

    各アプリのフォルダに`asp.bin`がビルドされる。

```
@startuml

salt
{
{T
    micro-ROS_ASP3
    + sample
    ++ joystick
    +++ **asp.bin**
    ++ ping_pong
    +++ **asp.bin**
    ++ publisher
    +++ **asp.bin**
    ++ subscriber
    +++ **asp.bin**
}
}

@enduml
```

micro-ROSの再ビルドの手順

1. 環境変数を設定

```

```

```
source /opt/ros/humble/setup.bash
source ~/uros_ws/install/local_setup.bash
```

Raspberry Pi 環境

エージェントとの通信

1. エージェントのビルド

   `Micro-XRCE-DDS-Agent`はLinuxのみでシリアル通信をサポートしているので、Raspberry Piでビルドした。

   下記の記事を参考に`Micro-XRCE-DDS-Agent`をビルドする。
   https://qiita.com/lutecia16v/items/5760551dd3a7a0d3e7d3

2. `Micro-XRCE-DDS-Agent`のコードをクローン

```
cd ~
git clone https://github.com/eProsima/Micro-XRCE-DDS-Agent.git
```

3. ビルド

```
cd Micro-XRCE-DDS-Agent
mkdir build && cd build
cmake -DTHIRDPARTY=ON ..
make
sudo make install
sudo ldconfig /usr/local/lib/
```

4. 実行

   `verbose_level`を6に設定して、メッセージの受信を表示するようにします。

```
MicroXRCEAgent serial --dev /dev/ttyACM0 -v 6
```

   USB-UARTの場合

```
MicroXRCEAgent serial --dev /dev/ttyUSB0 -v 6
```

   トピックが登録されていることを確認.

```

```

```
ros2 topic list
/microROS/ping
/microROS/pong
/parameter_events
/rosout
```

パブリッシュされているトピックを確認.

```
ros2 topic echo /microROS/ping
stamp:
  sec: 913
  nanosec: 145000000
frame_id: '412266125_323618167'
```

トピックを送る.

```
ros2 topic echo /microROS/pong
```

```
ros2 topic pub --once /microROS/ping std_msgs/msg/Header '{frame_id:
"fake_ping"}'
```