CADi
ROS as a Development Platform
Rosserial

Renato Galluzzi
renato.galluzzi@tec.mx

July 2020

What is Rosserial?

- Point-to-point communication protocol over a serial transmission line.
- Publish/Subscribe mechanism.
- It can exploit useful visualization tools in ROS, like Rviz or rqt.
- Widely supported for many platforms like:
  - Arduino
  - STM32 MCUs
  - Xbee
  - TI TivaC
  - Embedded Linux

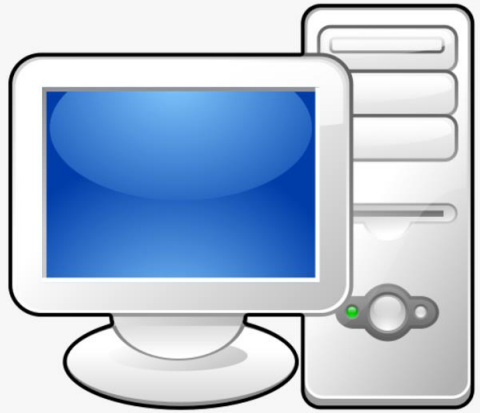
Packet Format

Packet bytes

1	2	3	4	5	6	7	N	N + 8
Sync Flag 0xFF	**Sync Flag / Protocol Version** 0xFE	**Message Length** LSB	**Message Length** MSB	**Message Length Checksum**	**Topic ID** LSB	**Topic ID** MSB	**Message Data**	**Topic and Data Checksum**

Serial communication

COM port



Installation (Rosserial for Arduino)

1. Install the following packages:

*Replace with your particular distribution

```
sudo apt-get install ros-melodic-rosserial-arduino
sudo apt-get install ros-melodic-rosserial
```

2. Clone the package git in your workspace and rebuild:

```
cd <ws>/src
git clone https://github.com/ros-drivers/rosserial.git
cd <ws>
catkin_make
catkin_make install
```

Installation (Rosserial for Arduino)

3. Install the Arduino IDE.

4. In the library manager, search for the last version of the Rosserial and install it.

This process has been already performed in your virtual machines!

Getting Started: Blink (Subscriber)

Objective

- Deploy a rosserial subscriber with Arduino. At each message reception from the host PC, the subscriber will toggle a LED.

Code (C++ in Arduino)

- Open File>Examples> Rosserial Arduino Library>Blink in the Arduino IDE.

Getting Started: Blink (Subscriber)

Before execution

- Enable your serial port in your virtual machine.
- Assign read and write permissions to the serial port in Ubuntu:

```
sudo usermod -a -G dialout user
sudo chmod a+rw /dev/ttyACM0
```

- This process must be repeated on each terminal session where you want to access the serial port.

Getting Started: Blink (Subscriber)

Execution

- Run `roscore`
- Run the rosserial_python package through

```
rosrun rosserial_python serial_node.py /dev/ttyACM0
```

- Publish an empty message on the “toggle_led” topic

```
rostopic pub toggle_led std_msgs/Empty --once
```

- You should see a LED toggle every time you send this message.

Poor Man's Oscilloscope

Objective

- Read the six analog inputs of Arduino and publish their values for visualization in the host PC.
- We will be using a custom message ADC.msg, which contains six uint16 values corresponding to the ADC readings.

Code (C++ in Arduino)

- Open File>Examples> Rosserial Arduino Library>ADC in the Arduino IDE.

Poor Man's Oscilloscope

Before execution

- Prepare your serial port as seen before.

Execution

- Run `roscore` and the rosserial_python package as seen before.
- Explore the topic adc/adcn: visualize through echo and rqt_plot. For instance:

```
rostopic echo adc/adc0
rqt_plot adc/adc0
```

Joystick-Controlled Servo