Vol 58, No 1—February 2024

ANALOG DEVICES | ADI Analog Dialogue™

Enabling Robot Operating Systems—Introducing the ADI Trinamic Motor Controller ROS1 Driver

Krizelle Paulene Apostol, Software Systems Engineer,
Jamila Macagba, Senior Software Systems Engineer, and
Maggie Maralit, Software Systems Design Engineering Manager

Abstract

Robot Operating System (ROS) drivers were developed on Analog Devices products so that they can be readily used within a ROS ecosystem. This article will give an overview on how to use and integrate these drivers in their applications, products, and systems (for example, autonomous navigation, safety bubble maps, and data collection robots); and how doing this will enable them to evaluate new technology immediately and avoid interoperability issues with third-party products. Among all products that will be discussed here, the focus will be on the recently released ROS driver for ADI Trinamic™ motor controllers—which are complete, board-level modules for embedded motion control, equipped with ADI Trinamic motion control expertise and ADI's analog process technology and power design skills.[1]

What Is ROS?

ROS is robotics middleware containing a set of software libraries and powerful developer tools from drivers to state-of-the-art algorithms—upon which robotic systems or applications can be developed. It is multidomain (for example, consumer, industrial, automotive, etc.), and supports multiple platforms that is, Linux, Windows, MacOS, and some embedded platforms)—plus it's 100% open source with commercial options. Support for ROS is abundant due to the dedicated resources from the global community, giving users an easier path for their designs and applications.

How Does the Technology Work?

ROS started in 2007 and became one of the most popular prototyping platforms for robotic development in fields such as self-driving cars, industrial robots, aerial vehicles, and more. It has continuously evolved and now has two versions: ROS1 and ROS2.

ROS1 and ROS2 systems must be isolated but the ROS bridge enables communication and exchange of data between them. More info is available at the ros2/ros1_bridge page.

Table 1. Major Differences of ROS1 and ROS2[2]

Factors	ROS1	ROS2
Communications protocol	XMLRPC + TCPROS	DDS
Architecture	ROS master + distributed	Fully decentralized
Build system	Catkin (cmake-based)	colcon/ament (cmake-based)
Build output	ros_ws/devel	ros_ws/install
Parameters	Global parameter server Dynamic reconfigure	Per-node parameters
Launch	XML	Python (+XML, YAML alternatives)
Commands	roslaunch, rosrun, rostopic, etc.	ros2 launch, ros2 run, ros2 topic, etc.
Platforms	Primarily ubuntu	Linux, MacOS, Windows

ROS Supported Platforms

ROS Noetic is the final version of ROS1 that will be supported until May 2025, while ROS2 has continuously rolling development distributions since its introduction in June 2020.

For a complete list, check these links for ROS1 supported platforms and ROS2 supported platforms.

ROS Basic Concepts

Some of the basic concepts of ROS, as shown in Figure 1, are packages, nodes, topics, services, and messages.

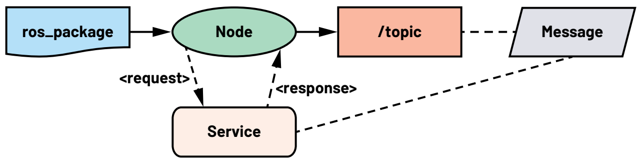

Figure 1. A ROS basic data flow.

Note: The ROS basic concepts discussed in the following paragraphs are similar for both ROS1 and ROS2.

Package

The ROS package is the main organization system of ROS programs or nodes. This is the most atomic build/release item in ROS. When creating a ROS package, it's important to set up a dedicated ROS workspace. This workspace is called the catkin workspace, wherein catkin is the official build system of ROS.

Nodes

ROS nodes are the executable programs made in ROS. They are processes that perform a specific task. ROS nodes can communicate with each other by use of ROS client libraries such as rospy, a Python client library, and *roscpp*, a C++ client library. Nodes can be a topic subscriber or/and a topic publisher. Nodes can also provide or use a service.[3]

Topics

ROS topics are the channels of the data being generated (or in ROS terms, published) by a ROS node.

In ROS, publisher nodes are the broadcasters of a topic, while subscriber nodes are the listeners of a topic.

In Figure 2, generic_motor_control's node is the broadcaster. /cmd_vel is the topic that velocity_publisher publishes. This means that velocity_publisher provides velocity information based on motor control (or command velocity).

While ros_application's node is the listener, velocity_subscriber subscribes to the topic /cmd_vel. This means that velocity_subscriber accesses or uses the velocity information provided by velocity_publisher.

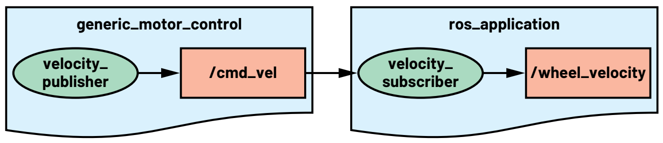

Figure 2. Publisher-subscriber.

Messages

While topics are the data channels, messages are the data in ROS-compatible format applicable to different sensors.

The following are sample sensors and applicable to the ROS message format:

- Time of flight (ToF) cameras: sensor_msgs/Image, sensor_msgs/PointCloud
- Inertial measurement unit (IMU) sensor: sensor_msgs/Imu
- Motor control: geometry_msgs/Twist
- Wheel encoders: geometry_msgs/TwistStamped, geometry_msgs/TwistWithCovarianceStamped

ROS topics communicate by sending messages (topic publisher) or receiving messages (topic subscriber) and must be on a matching data type.

For example, in Figure 2, the velocity information (command velocity) from the velocity_publisher node wants to be accessed/used by the velocity_subscriber node. If the topic publisher, velocity_publisher, uses data type geometry_msgs/Twist, the topic subscriber, velocity_subscriber, should have the same data type as well.

Services

The publisher-subscriber communication model is open-ended and not applicable to reply interactions that are often required in a distributed system.[4]

Services are allowing nodes to communicate by sending a request and receiving a response. The publisher-subscriber communication model uses .srv files wherein service descriptions, such as message type of request and response, are specified.

Service is a two-way synchronous communication model wherein there is a client and a server. The server node is the one that provides the service while the client node is the one that sends the request and waits for a response from the server node.

For example, in Figure 3, server_node provides a service, SetVelocity.srv, to change command velocity, *vel*. The service accepts the velocity value in float32 format and returns the status in string format; "success" if the requested velocity was set or "FAIL" if not.

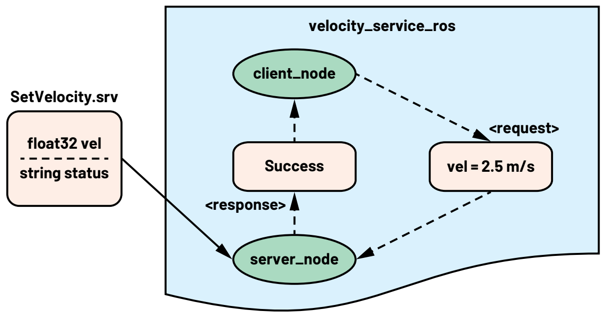

Figure 3. Sample use of service.

The client_node sends a request to set the command velocity to 2.5 m/s. Once server_node receives the request, it sends the response of "success".

Integrating ADI Solutions into the ROS Ecosystem

ADI is an official member of the ROS-Industrial Consortium—an open-source project that extends the advanced capabilities of ROS software to industrial-relevant hardware and applications.[5] Being a part of this community, ADI has initially targeted dedicated modules geared toward the industrial domain.

ADI has developed ROS drivers for different dedicated modules. To showcase the developed drivers and leverage the capabilities of ROS, ADI developed the Analog Devices Autonomous Mobot (ADAM) as an in-house autonomous mobile platform (see Figure 4).

Figure 4. The ADAM.

ADAM: Analog Devices Autonomous Mobot

ADAM is powered by ROS and is enabled by different ROS-powered devices. The platform shows how ADI's ROS drivers can be integrated into mobile robot applications—specifically autonomous navigation.

Figure 5 shows the high level hardware diagram of ADAM with different modules. It mainly has the following devices connected:

- ADIS16470 or IMU sensors based on multiaxis combinations of precision gyroscopes, accelerometers, magnetometers, and pressure sensors—which are primarily used as sensing feedback to improve position/direction estimation.
- ADBMS6948, a multicell battery monitor, measuring up to 16 series-connected battery cells with very high measurement accuracy over the entire temperature range.
- EVAL-ADTF3175D-NXZ or CMOS ToF camera offering the highest resolution in the market, and can be complemented with depth computation and processing, laser drivers, power management, and development tools with reference firmware/software.
- ADI Trinamic motor controllers, which are complete, board-level solutions for embedded motion control, equipped with ADI Trinamic motion control expertise and ADI's analog process technology and power design skills.[1]

Figure 6 shows the high level ROS architecture of the ADAM using ROS drivers and multiple application/algorithm nodes needed for autonomous navigation. The IMU data (*/imu/data_raw*) and ADI Trinamic motor controller feedback (*/tmc_info*) are used as inputs for pose estimation, resulting in the robot's odometry (*/odom*). The lidar data (/scan) is the primary input for the simultaneous localization and mapping (SLAM) algorithm for generating a map; ToF data (*/image_raw*) can also be used as input in other SLAM algorithms. The move_base node will then wait for any goal pose from the user and send velocity commands (*/cmd_vel*) to the ADI Trinamic motor controller to move the robot.

ADI Trinamic Motor Controller ROS Driver

ADI Trinamic motor controllers (TMCs) are complete, board-level solutions for embedded motion control, equipped with ADI Trinamic motion control expertise as well as ADI's analog process technology and power design skills.[1] Supporting various types of motors like single-/multiple-axis stepper and brushless DC (BLDC), the available interfaces include CAN, EtherCAT®, RS-232, RS-485 and USB, and protocols supported range from Trinamic Motion Control Language (TMCL™), CANopen®, over EtherCAT (CoE), CANopen or Modbus.[6]

An IDE called TMCL-IDE is available to assist the users to develop applications and allow easy reprogramming of these modules. It uses either TMCL for standalone operation or the standardized CANopen® protocol, and it allows users to set parameters, visualize data in real-time, and develop/debug standalone applications.

As TMCs enable a new class of intelligent actuators and as ROS becomes more and more prevalent, especially in robotics, additional support like ROS drivers for these modules were developed to extend use cases even further for manufacturing and industrial automation. Specifically, these ROS drivers will be expected to:

- Control the motors' velocity, position, or torque
- Monitor motor controller and motor information

The TMC ROS driver is similar to what TMCL-IDE offers except that it enables nodes from a ROS-capable system to readily use these TMCs without installing any additional drivers. At the time this article is published, it is only able to support CAN interface (specifically SocketCAN), although other interfaces are ongoing development and planned to be supported soon.

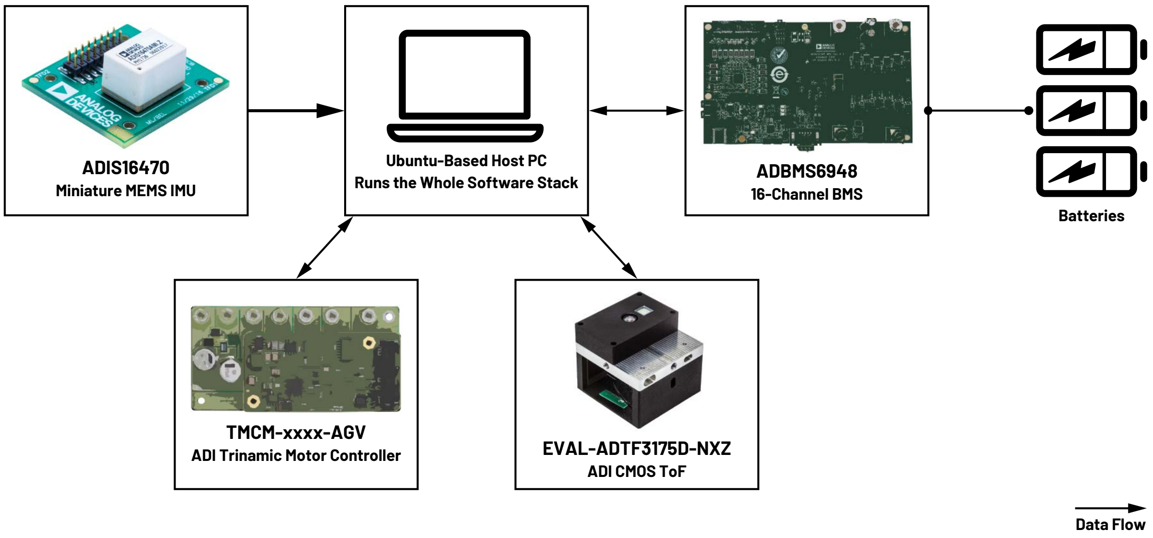

Figure 5. A high level hardware diagram of the ADAM.

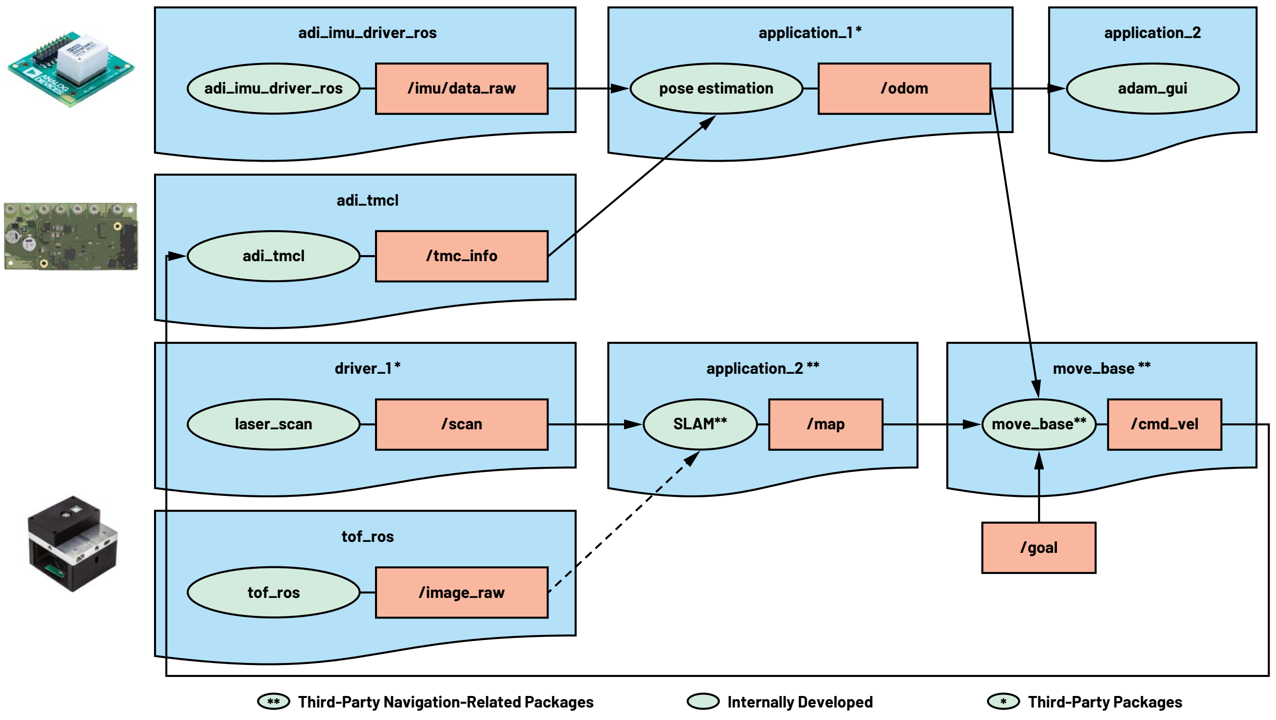

Figure 6. A high level ROS architecture of the ADAM's navigation stack.

The ADI Trinamic motor controller modules (TMCM) currently supported are listed here.

Software Architecture

Figure 7 shows the high level software architecture of the adi_tmcl.

As seen in Figure 7, the adi_tmcl does not need any additional drivers because it makes use of SocketCAN drivers already supported by default in most Linux-based systems. Additionally, adi_tmcl has its own TMCL protocol interpreter, which makes it able to understand TMCL-compliant send/receive commands requested by the user. As the last layer, the tmcl_ros_node provides the direct interface on the ROS system in the form of publishers, subscribers, and services. Each of these offer certain functionalities configurable with a group of parameters detailed in the following sections.

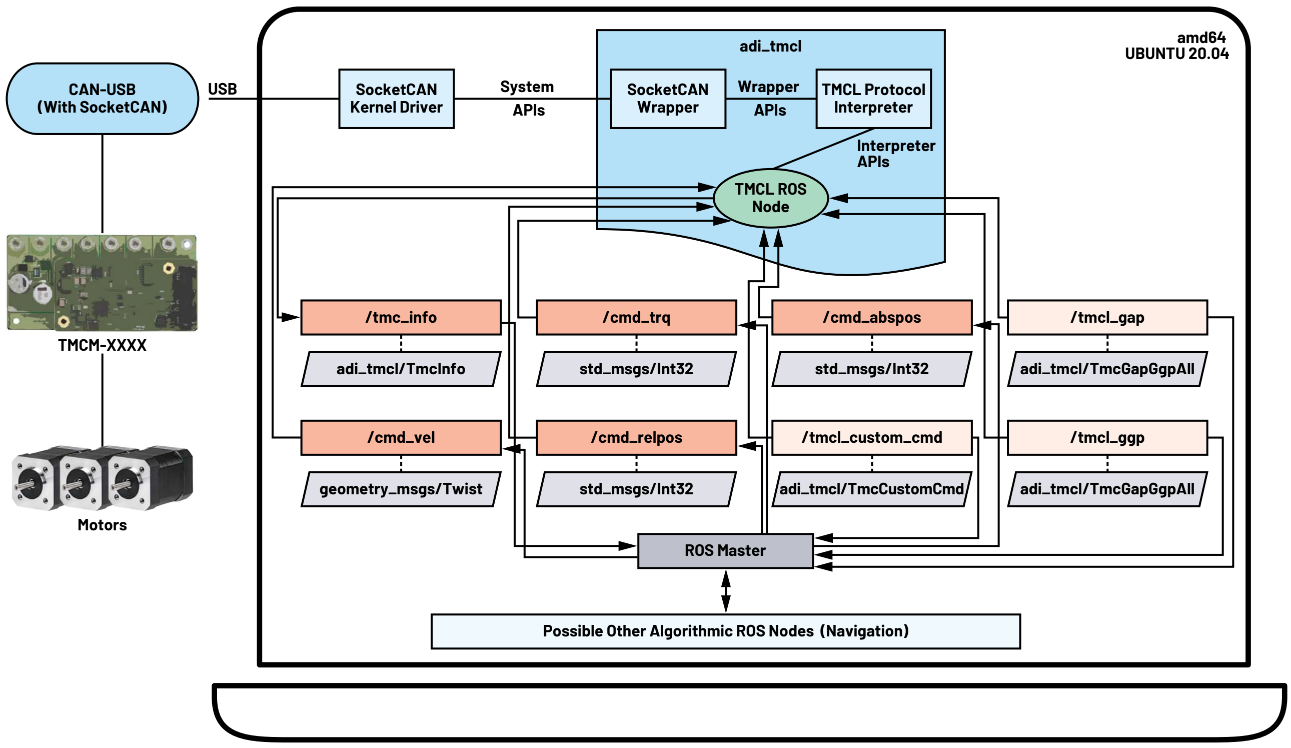

Figure 7. A high level software architecture of adi_tmcl.

Features

The adi_tmcl offers a range of features including:

1. Support for different TMC boards
2. One-time configuration of TMC modules using TMCL-IDE
3. A move/stop motor
4. The ability to get TMC/motor information
5. Execute custom TMC commands
6. The ability to get all axis parameter values
7. The ability to get all global parameter values
8. Support for multiple TMC board setup
9. Easy integration into ROS systems/applications

Stay tuned for an article in next month's issue of *Analog Dialogue*, "Mastering ADI Trinamic Motor Controllers with the ROS1 Driver," which discusses the details of these features with samples on how to utilize them.

Conclusion

ADI Trinamic motor controllers enable a new class of intelligent actuators. As ROS became more and more prevalent, especially in robotics, additional support for these modules, like ROS drivers, was developed to extend use cases even further for manufacturing and industrial automation.

In this article, we showed how ROS extends devices to have:

- Added value, like extendibility to industrial applications;
- Easier interoperability with third-party products made possible by the ROS communications framework;
- Wider options for customers to leverage ADI products in their systems; and
- The ability to evaluate new technology quickly and start using immediately.

For more information, visit ADI's Industrial Robotics page.

References

[1]"ADI Trinamic Hardware for Motor and Motion Control." Analog Devices, Inc.

[2]"ros2/ros2_documentation." GitHub, Inc.

[3]"Understanding ROS Nodes." ROS.org.

[4]"Services." ROS.org.

[5]"ROS-Industrial." ROS.org.

[6]"Industrial Communication Protocols and Interfaces for Motion Control Applications." Analog Devices, Inc.

What's Next?

- Watch out for a deep-dive article on ADI Trinamic motor controller ROS1 driver
- Watch out for a future article on ROS2 for ADI Trinamic motor controller
- Download the ADI Trinamic motor controller ROS1 and ROS2 driver
- Purchase ADI Trinamic motors and motor controller evaluation boards

About the Author

Krizelle Paulene Apostol is a software systems engineer and part of the Philippine Development Center working with the Intelligent Motion and Robotics Group at Analog Devices. She joined ADI in Cavite, Philippines in December 2019. She graduated from FAITH Colleges with a bachelor's degree in computer engineering. She has been involved in various projects that focus on ROS, Gazebo simulations, firmware development, communication protocols, and algorithm development.

About the Author

Jamila "Jam" Aria Macagba is a senior software systems engineer and part of the Philippine Development Center working with the Intelligent Motion and Robotics Group at Analog Devices. She joined ADI in Cavite, Philippines in July 2018 and graduated from University of the Philippines Los Baños with a bachelor's degree in electrical engineering. Her focus is on ROS driver development and integration in ROS systems.

About the Author

Maggie is a software systems design engineering manager and part of the Philippine Development Center working with the Industrial Motion and Robotics Group at Analog Devices, Inc. She joined ADI in Cavite, Philippines in April 2019. She graduated from University of the Philippines in Los Baños, Laguna with a bachelor's degree in computer science. She currently leads a group of engineers in the Philippines site supporting projects on industrial robotics. Maggie previously worked as an application specialist at Hewlett-Packard from 2009 to 2010; senior software engineer at Canon Information Technologies Phils., Inc. from 2010 to 2013; firmware development engineer at Ionics EMS, Inc. from 2013 to 2015; and senior embedded software engineer at Continental Automotive Singapore from 2015 to 2019.

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

For regional headquarters, sales, and distributors or to contact customer service and technical support, visit analog.com/contact.

Ask our ADI technology experts tough questions, browse FAQs, or join a conversation at the EngineerZone Online Support Community. Visit ez.analog.com.

©2024 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners.

VISIT ANALOG.COM