ANALOG DEVICES 超越一切可能™ | ADI AnalogDialogue

使用ROS1驱动程序来操控ADI Trinamic电机控制器

Krizelle Paulene Apostol，软件系统工程师
Jamila Macagba，高级软件系统工程师
Maggie Maralit，软件系统设计工程经理

摘要

“实现机器人操作系统——电机控制器ROS1驱动程序简介”一文中概述了新型ADI Trinamic™电机控制器(TMC)驱动程序，并讨论了将电机控制器集成到机器人操作系统(ROS)生态系统中的方法。TMC ROS1驱动程序支持TMC驱动层和应用层之间在ROS框架内无缝通信，且适用于它支持的各种TMC板。本文将深入探讨TMC ROS1驱动程序的功能，包括电机控制、信息检索、命令执行、参数获取以及对多种设置的支持。文中还概述了如何将电机控制器集成到嵌入式系统和应用中，从而利用ROS框架提供的优势。

ADI Trinamic电机控制器ROS1驱动程序

ROS是一个机器人系统中间层，包含一组软件库和强大的开发工具，从驱动程序到最先进的算法，可以在此基础上开发机器人系统或应用程序。ADI Trinamic电机控制器支持新型智能执行器，并且随着ROS变得越来越流行，尤其是在机器人领域，为了扩展在制造和工业自动化应用中的适用性，我们开发了ROS驱动程序等附加模块支持。ADI公司的TMC ROS1驱动程序提供与Triaminic电机控制语言集成开发环境(TMCL-IDE)类似的功能，但有一个关键区别：它允许支持ROS的系统中的节点使用TMC，而无需额外安装驱动程序。此外，adi_tmcl集成了自己的TMCL协议解释器，因此能够解释符合TMCL标准的用户请求的命令。最后一层是tmcl_ros_node，它建立了与ROS系统的直接接口，提供发布者、订阅者和服务等功能。每一个功能都可以使用一组参数进行自定义，以下部分将详细讨论这些功能。

功能

1. 支持多种TMC模块

TMC ROS驱动程序或adi_tmcl旨在支持所有遵守TMCL协议的商用TMC。截至本文发布，它目前支持CAN接口（特别是SocketCAN）。但开发工作还在进行，不久的将来会支持其他接口。这些TMC包含ADI Trinamic PANdrive™智能电机和模块，可以支持步进电机和直流无刷伺服(BLDC)电机。由于使用ROS参数，adi_tmcl能够无缝支持不同的TMC模块。只需配置**tmcl_ros_node**而无需重新构建整个控制包。

在adi_tmcl/config目录中，每个ADI Trinamic电机控制器模块(TMCM)都有两个相关的YAML文件。这些文件以人类可读的数据序列化语言编写，包含ROS参数，应在执行期间加载：

- adi_tmcl/config/autogenerated/**TMCM-XXXX.yaml**

此YAML文件是自动生成的，包含特定于模块的参数，不建议修改，以免导致节点行为异常。

- adi_tmcl/config/**TMCM-XXXX_Ext.yaml**

此YAML文件包含用户可以修改的所有参数，以便(1)与板通信（例如接口名称），(2)实现电机控制，以及(3)更改ROS主题名称。

例如，如果您想使用TMCM-1636（图3），只需运行图1所示的代码。

Terminal

```
# Launch using TMCM-1636
~/catkin_ws $ roslaunch adi_tmcl tmcm_1636.launch
# To exit the node, press Ctrl + C
```

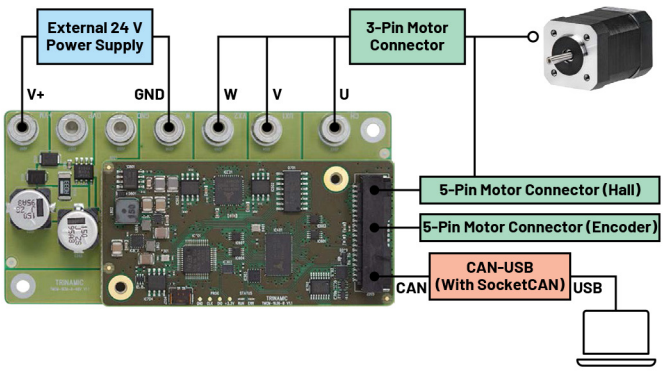
图1. 启动TMCM-1636。

其中，adi_tmcl/launch/tmcm_1636.launch加载TMCM-1636专用的YAML文件。

Tmcm_1636.launch

```
[…]
    <!—Launches node→
     <node name="tmcl_ros_node" pkg="adi_tmcl" type="tmcl_ros_node"
     output="screen" required="true">
      <!-- Autogenerated YAML file containing TMCM-1636
      configurations -->
       <rosparam command="load" file="$(find adi_tmcl)
       /config/autogenerated/TMCM-1636.yaml" />
      <!-- User-generated YAML file containing ROS-specific parameters
      as well as user-set values for TMCM-1636 configurations -->
       <rosparam command="load" file="$(find adi_tmcl)/config/
       TMCM-1636_Ext.yaml" />
</node>
[…]
```

图2. 使用TMCM-1636运行TMC ROS驱动程序的代码片段。



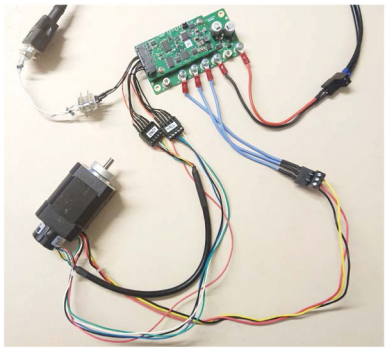


图3. （上）TMCM-1636硬件连接图；（下）实际设置的参考图片。

要使用TMCM-1260（图6），请运行以下命令：

Terminal

```
# Launch using TMCM-1260
~/catkin_ws $ roslaunch adi_tmcl tmcm_1260.launch
# To exit the node, press Ctrl + C
```

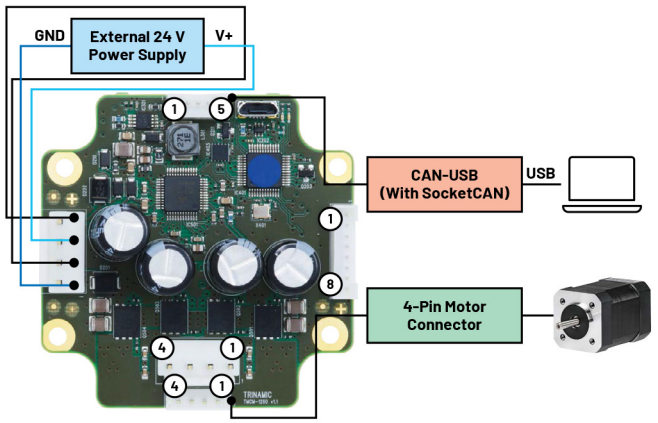
图4. 使用TMCM-1260启动TMC ROS驱动程序的命令。

其中，adi_tmcl/launch/tmcm_1260.launch加载TMCM-1260专用的YAML文件。

tmcm_1260.launch

```
[…]
    <!--Launches node -->
     <node name="tmcl_ros_node" pkg="adi_tmcl" type="tmcl_ros_node"
     output="screen" required="true">
      <!-- Autogenerated YAML file containing TMCM 1260
      configurations -->
       <rosparam command="load" file="$(find adi_tmcl)/config/
       autogenerated/TMCM-1260.yaml" />
      <!-- User-generated YAML file containing ROSspecific parameters
      as well as user-set values for TMCM-1260 configurations -->
       <rosparam command="load" file="$(find adi_tmcl)/config/
       TMCM-1260_Ext.yaml" />
</node>
[…]
```

图5. 使用TMCM-1260运行TMC ROS驱动程序的代码片段。



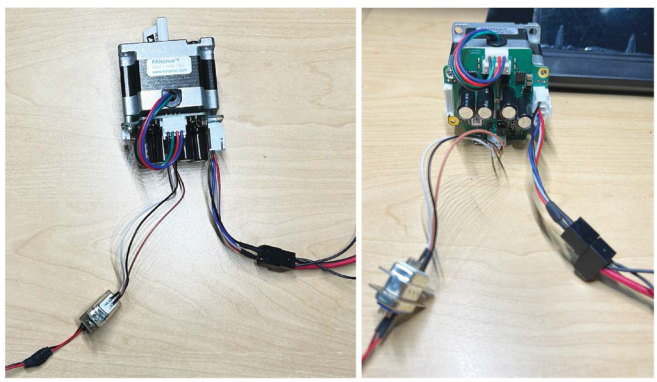


图6. （上）TMCM-1260硬件连接图；（下）实际设置的参考图片。

启动目录包括所有支持的TMC模块，可以点击此处查看。

2. 使用TMCL-IDE一次性配置TMC模块

在通过ROS使用TMC模块之前，需要根据所使用的电机完成配置。所有的配置使用TMCL-IDE完成，并应存储在EEPROM中（否则可能无法正确控制电机）。

- BLDC电机模块（如TMCM-1636）
  - 有关如何在TMCL-IDE中通过Wizard Pool工具完成电机校准的流程/教程，请查看此教程。
  - 有关如何在TMCL-IDE中完成比例积分(PI)调谐功能的流程/教程，请查看此教程。
- 步进电机模块（如TMCM-1260）
  - 有关如何在TMCL-IDE中通过Wizard Pool功能完成初始化配置的流程/教程，请查看此教程。

初始化和调谐后，务必将所有参数存储在板的EEPROM中。这可以通过如下方法来完成：(1) store参数，(2) STAP命令，以及/或者(3)创建和上传TMCL程序并启用自动启动模式。有些板仅支持其中的少数选项。

TMC ROS驱动程序的设计得到了简化，在完成TMC模块和电机的初始化配置/调谐后，基于使用TMCL-IDE的一次性配置即可控制电机。

3. 移动/停止电机

TMC ROS驱动程序通过在以下任一主题中发布命令来移动/停止电机：

- */cmd_vel* (geometry_msgs/Twist)—设置电机转速
- */cmd_abspos* (std_msgs/Int32)—设置电机的绝对位置
- */cmd_relpos* (std_msgs/Int32)—设置电机的相对位置
- */cmd_trq* (std_msgs/Int32)—设置电机扭矩

注：多轴TMC设置中的不同电机有不同的地址。

用户可以连接ROS系统来发送至这些特定指令，从而控制电机的运动。指令的选择取决于具体应用、TMC设置以及所用电机的类型。例如，对于轮式机器人，用户可以选择设置速度，而对于夹具，设置位置会更合适。

作为说明性示例，可以看看脚本adi_tmcl/scripts/fake_cmd_vel.sh。这个简单的脚本可以控制电机以顺时针和逆时针两个方向旋转，并且逐渐提高转速。要执行此脚本，请按照图7所示的命令进行操作。

Terminal #1

```
~ $ cd ~/catkin_ws
~/catkin_ws $ source /opt/ros/noetic/setup.bash
~/catkin_ws $ source devel/setup.bash
~/catkin_ws $ roslaunch adi_tmcl tmcm_1260.launch
        # or $ roslaunch adi_tmcl tmcm_1636.launch
```

Terminal #2

```
~ $ cd ~/catkin_ws
~/catkin_ws $ source /opt/ros/noetic/setup.bash
~/catkin_ws $ source devel/setup.bash
~/catkin_ws $ rostopic echo /tmc_info_0
```

Terminal #3

```
~ $ cd ~/catkin_ws/src/adi_tmcl/scripts
~/catkin_ws/src/adi_tmcl/scripts $ sudo chmod +x fake_cmd_vel.sh
~/catkin_ws/src/adi_tmcl/scripts $ ./fake_cmd_vel.sh
```

图7. 用于测试TMC ROS驱动程序转速控制的命令。

注意：

- 2号终端窗口和3号终端窗口最好并排显示。
- 可以按Ctrl-C复制1号终端窗口中的命令，完成后粘贴到2号终端窗口中。
- 3号终端窗口中的命令会自行停止。

为了验证电机是否已移动，图8显示了来自TMC */tmc_info_0*的实际转速反馈图。

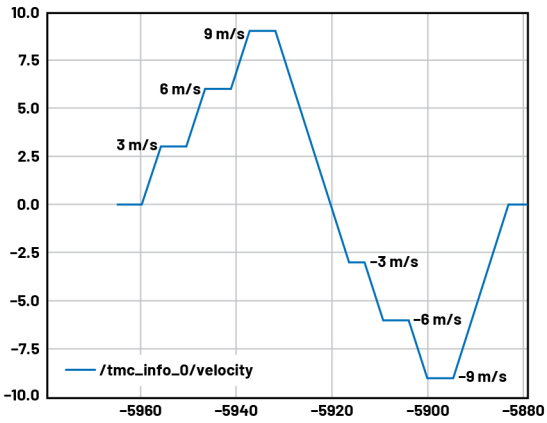


图8. 使用RQT绘制的电机实际转速图（以m/s为单位）。

4. TMC/电机信息检索

系统可以通过订阅以下主题，从TMC ROS驱动程序检索信息：

- */tmc_info (adi_tmcl/TmcInfo)* - 提供电压、TMC状态、实际转速、实际位置和实际扭矩信息

注：多轴TMC设置中的不同电机有不同的主题。

用户可以链接ROS系统来订阅这些指定的主题。这样，用户就可以监视参数值，并根据参数值采取行动。例如，在特定于应用的场景中，当检测到TMC状态出错时，操作员可能会选择停止系统，或者在电机到达特定位置时执行预编程的动作。

作为例子，adi_tmcl/scripts/fake_cmd_pos.sh是一个简单的脚本，它让电机先顺时针旋转，再逆时针旋转，并且不断提高位置幅度。请执行图9所示的命令。

```
Terminal #1
~ $ cd ~/catkin_ws
~/catkin_ws $ source /opt/ros/noetic/setup.bash
~/catkin_ws $ source devel/setup.bash
~/catkin_ws $ roslaunch adi_tmcl tmcm_1260.launch
        # or $ roslaunch adi_tmcl tmcm_1636.launch

Terminal #2
~ $ cd ~/catkin_ws
~/catkin_ws $ source /opt/ros/noetic/setup.bash
~/catkin_ws $ source devel/setup.bash
~/catkin_ws $ rostopic echo /tmc_info_0

Terminal #3
~ $ cd ~/catkin_ws/src/adi_tmcl/scripts
~/catkin_ws/src/adi_tmcl/scripts $ sudo chmod +x fake_cmd_pos.sh
~/catkin_ws/src/adi_tmcl/scripts $ ./fake_cmd_pos.sh
```

图9. 用于测试TMC ROS驱动程序位置控制的命令。

为了验证电机是否已移动，图10显示了来自TMC (*/tmc_info_0*)的实际位置回读图。

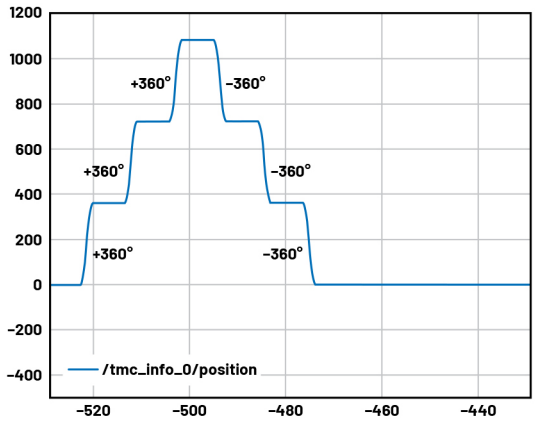


图10. 使用RQT绘制的电机实际位置图（以度为单位）。

5. 执行自定义TMC命令

系统可以通过执行以下功能来访问和调整TMC参数：

- tmcl_custom_cmd (adi_tmcl/TmcCustomCmd) - 获取/设置TMC的轴参数AP和全局参数(GP)的值

用户可以选择将此服务集成到ROS系统中，以满足特定应用需求。此功能使用户能够直接从ROS驱动程序配置TMC板。例如，用户可以选择设置轴参数(SAP)以获得最大电流，从而调整允许的绝对电流水平。但是，用户必须透彻了解他们要通过此功能修改的参数，不正确的设置可能会导致TMC ROS驱动程序故障。因此，强烈建议任何配置都通过TMCL-IDE执行。图11提供了调用此服务的示例，展示了使用指令类型208对DrvStatusFlags进行获取轴参数(GAP)操作。

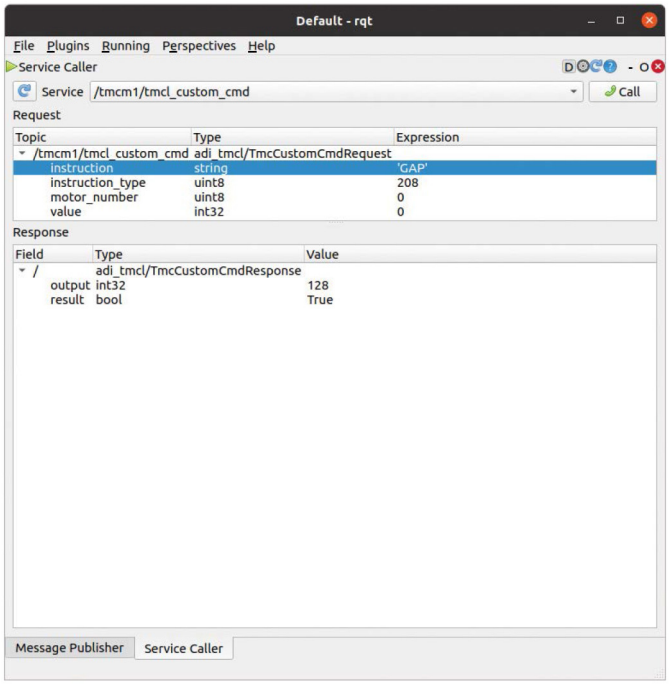


图11. 通过RQT触发的tmcl_custom_cmd服务。

6. 访问所有轴参数值

系统可以通过以下方式访问TMC轴参数值：

- tmcl_gap (adi_tmcl/TmcGapGgpAll) - 获取指定电机/轴的所有TMC轴参数(AP)的值

用户可以将ROS系统与此功能集成，以满足特定应用的需求。例如，此服务可以捕获TMC板的当前设置和状态，包括AP（例如编码器步长、PI调谐、换向模式等）。

图12显示了部分输出示例。通过分析该结果，用户可以确认一次性配置是否正确保存在板的EEPROM中。

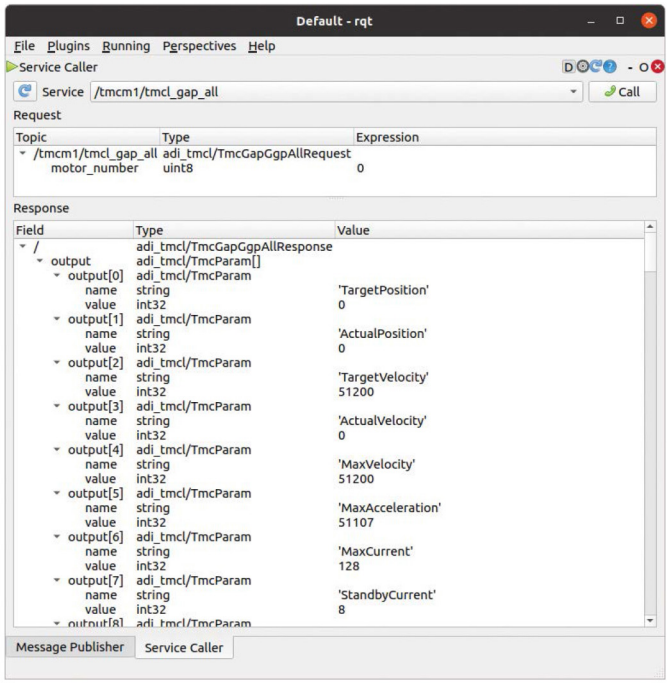


图12. 通过RQT触发的tmcl_gap_all服务。

7. 访问所有全局参数值

系统可以通过以下方式访问TMC全局参数值：

- tmcl_ggp (adi_tmcl/TmcGapGgpAll) - 获取所有TMC全局参数(GP)的值

此功能可以检索TMC板的当前配置和状态。可访问的一些GP包括：CAN比特率、串行波特率、自动启动模式等。

图13显示了执行此服务后获得的部分输出。此结果使用户能够确认一次性配置是否已正确存储在板的EEPROM中。

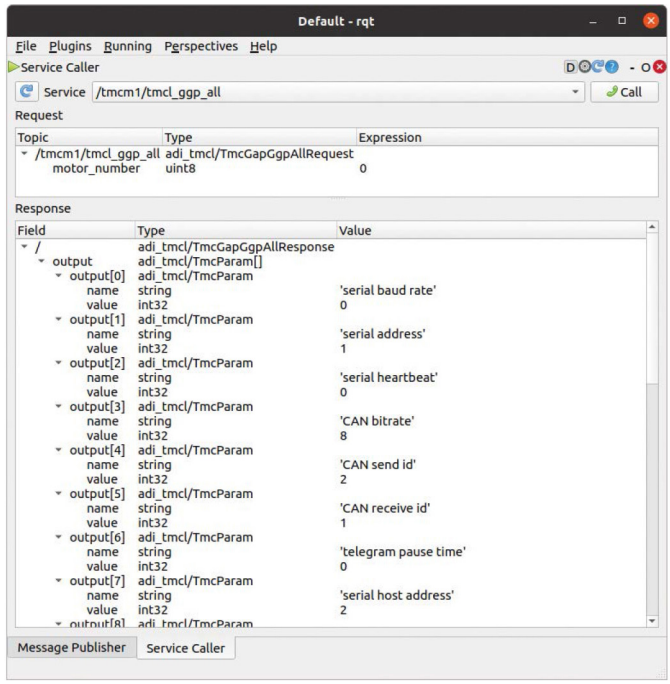


图13. 通过RQT触发的tmcl_ggp_all。

8. 多个TMC板设置

对于可能需要多个TMC模块的较大系统（如机械臂），TMC ROS驱动程序支持多个器件设置。

a. 多个CAN通道中的多个TMC板

如图14所示，当用户的每个TMC板都有一个CAN-USB时，系统将添加*命名空间*以区分每个节点的实例。在此特定用例中，需要相应更新**comm_interface_name**参数，以确保与板正确通信。

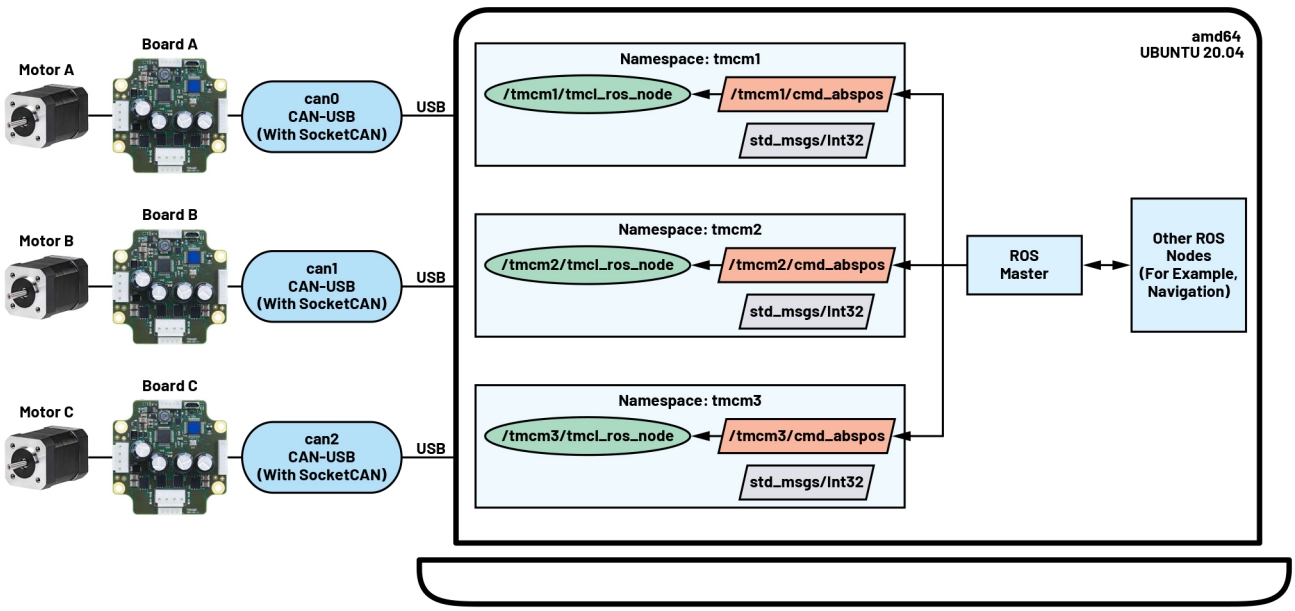


图14. 多个CAN通道中的多个TMC板的示例图。

multiple_tmcm_multiple_can_channel.launch

```
[…]
<group ns="tmcm1">
    <node name="tmcl_ros_node" pkg="adi_tmcl" type="tmcl_ros_node" output="screen" required="true">
     <rosparam command="load" file="$(find adi_tmcl)/config/autogenerated/TMCM-1260.yaml"/>
     <rosparam command="load" file="$(find adi_tmcl)/config/TMCM-1260_Ext.yaml" />
     <param name="comm_interface_name" type="string" value="can0" />
    </node>
</group>
<group ns="tmcm2">
    <node name="tmcl_ros_node" pkg="adi_tmcl" type="tmcl_ros_node" output="screen" required="true">
     <rosparam command="load" file="$(find adi_tmcl)/config/autogenerated/TMCM-1260.yaml"/>
     <rosparam command="load" file="$(find adi_tmcl)/config/TMCM-1260_Ext.yaml" />
     <param name="comm_interface_name" type="string" value="can1"/>
    </node>
</group>
<group ns="tmcm3">
    <node name="tmcl_ros_node" pkg="adi_tmcl" type="tmcl_ros_node" output="screen" required="true">
     <rosparam command="load" file="$(find adi_tmcl)/config/autogenerated/TMCM-1260.yaml"/>
     <rosparam command="load" file="$(find adi_tmcl)/config/TMCM-1260_Ext.yaml" />
     <param name="comm_interface_name" type="string" value="can2"/>
    </node>
</group>
[…]
```

图15. 使用多个CAN通道运行多个TMC ROS驱动程序的代码片段。

图15中的代码是用于设置此用例的示例启动文件。在此示例中，电机A可以通过发布到/*tmcm1*/cmd_abspos来控制，电机B可以通过发布到/*tmcm2*/cmd_abspos来控制，电机C可以通过发布到/*tmcm3*/cmd_abspos来控制。

b. 单个CAN通道中的多个TMC板

TMC ROS驱动程序支持的另一种设置是单个CAN通道中有多个TMC板，如图16所示。与上文所述的对多个TMC板的支持非常相似，系统引入命名空间来区分每个节点实例。所有板的**comm_interface_name**保持一致。调整**comm_tx_id**和**comm_rx_id**以确保与各板正确通信。

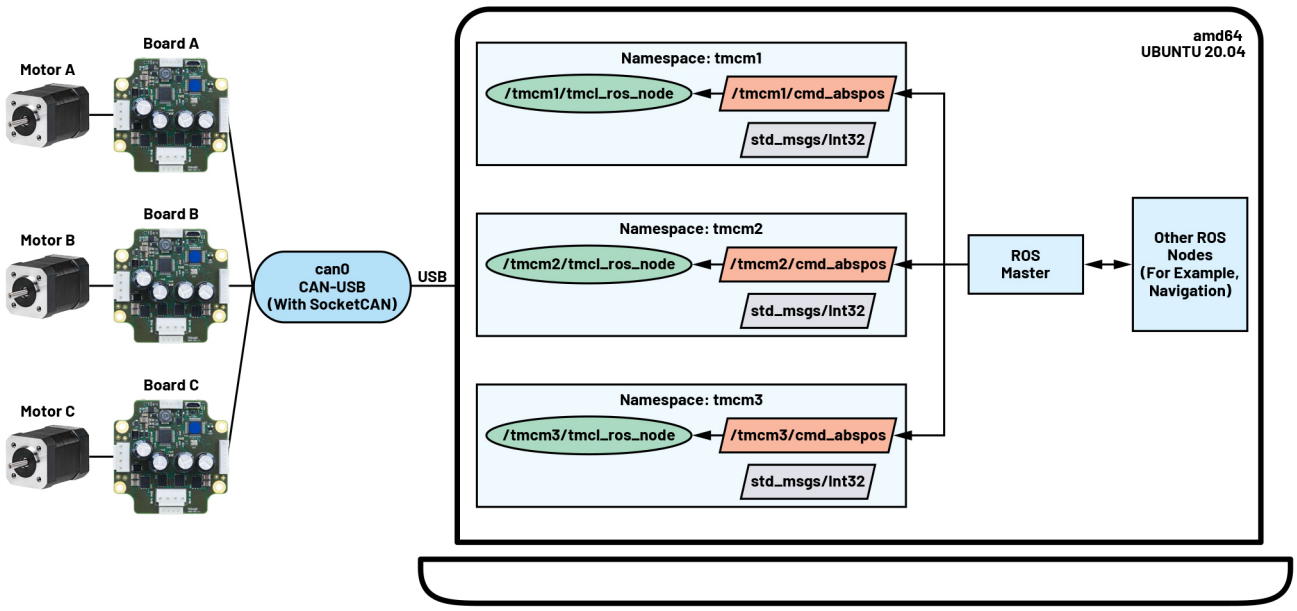


图16. 单CAN通道中的多个TMC板的示例图。

图17显示了用于设置此用例的示例启动文件。在此示例中，电机A可以通过发布到/*tmcm1*/*cmd_abspos*来控制，电机B可以通过发布到/*tmcm2*/*cmd_abspos*来控制，电机C可以通过发布到/*tmcm3*/*cmd_abspos*来控制。

9. 轻松集成到ROS系统/应用中

借助ROS提供的消息传递系统，即便是较大的系统也可以轻松地交换节点（例如驱动程序、算法等）。TMC ROS驱动程序将这一优势扩展到了TMC板，允许它无缝集成到ROS系统/应用中。

multiple_tmcm_single_can_channel.launch

```
[…]
<group ns="tmcm1">
    <node name= "tmcl_ros_node" pkg="adi_tmcl" type="tmcl_ros_node" output="screen" required="true">
     <rosparam command="load" file="$(find adi_tmcl)/config/autogenerated/TMCM-1260.yaml" />
     <rosparam command="load" file="$(find adi_tmcl)/config/TMCM-1260_Ext.yaml" />
     <param name="comm_tx_id" type="int" value="1"/>
     <param name="comm_rx_id" type="int" value="2"/>
    </node>
</group>
<group ns="tmcm2">
    <node name="tmcl_ros_node" pkg="adi_tmcl" type="tmcl_ros_node" output="screen" required="true">
     <rosparam command="load" file="$(find adi_tmcl)/config/autogenerated/TMCM-1260.yaml" />
     <rosparam command="load" file="$(find adi_tmcl)/config/TMCM-1260_Ext.yaml" />
     <param name="comm_tx_id" type="int" value="3"/>
     <param name="comm_rx_id" type="int" value="4"/>
    </node>
</group>
<group ns="tmcm3">
    <node name="tmcl_ros_node" pkg="adi_tmcl" type="tmcl_ros_node" output="screen" required="true">
     <rosparam command="load" file="$(find adi_tmcl)/config/autogenerated/TMCM-1260.yaml" />
     <rosparam command="load" file="$(find adi_tmcl)/config/TMCM-1260_Ext.yaml" />
     <param name="comm_tx_id" type="int" value="5"/>
     <param name="comm_rx_id" type="int" value="6"/>
    </node>
</group>
[…]
```

图17. 使用单个CAN通道运行多个TMC ROS驱动程序的代码片段。

a. 集成到AGV/AMR中

图18说明了**navigation_node**如何通过发送geometry_msg/Twist格式的*/cmd_vel*来控制移动机器人。然后，**motor_controller**通过Geometry_msg/Twist格式的*/wheel_velocity*发送反馈，使得**navigation_node**可以相应地重新校准。

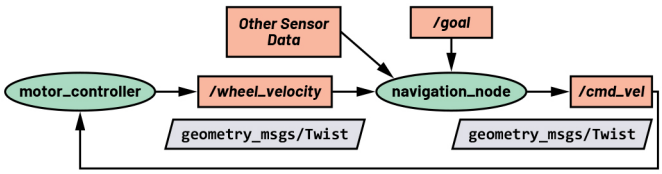


图18. AGV/AMR的简化架构。

通过了解**navigation_node**发布和订阅的位置，**tmcl_ros_node**可以轻松更改**motor_controller**（图19）。与TMC信息检索功能类似，adi_tmcl会发布关于车轮转速的实时信息，**wheel_velocity_node**会将车轮转速信息从adi_tmcl/TmcInfo转换为geometry_msg/Twist。由于新架构及其集成的TMC板符合正确的数据格式，因此移动机器人预计以相同方式工作。

b. 集成到机械臂中

图20说明了为将TMC板集成到采用机械臂的贴片应用中，控制机械臂需要使用多个电机。与之前的用例类似，用户需要确保**pick_and_place_node**会订阅/发布所预期的数据格式。

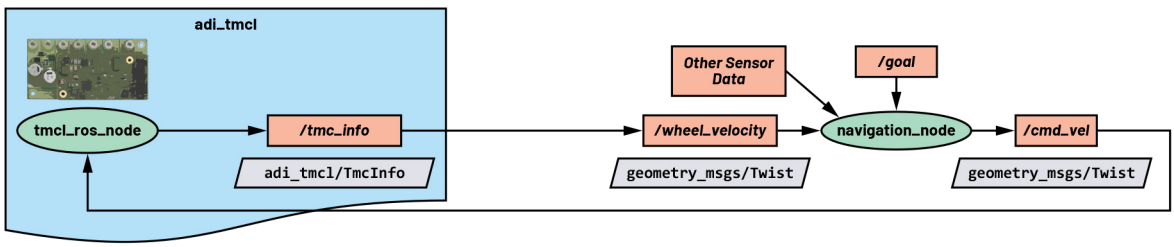


图19. 带有TMC ROS驱动程序的AGV/AMR简化架构。

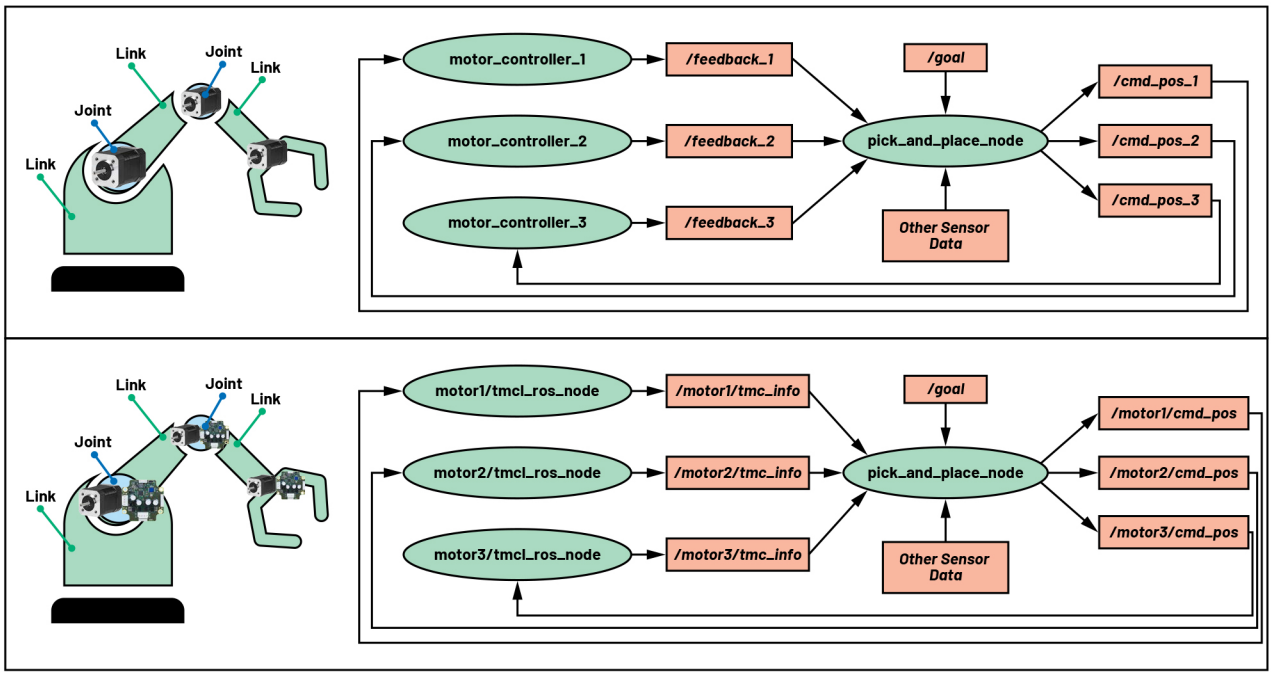


图20.（上）带有通用电机控制器的机械臂；（下）带有TMC板的机械臂。

有关将TMC板集成到ROS系统的分步指南以及如何利用所述的功能，请点击此处。

结论

ADI公司的TMC ROS1驱动程序支持TMC基础驱动层和应用层之间在ROS管理的系统内无缝通信，且适用于它支持的各种TMC模块。

本文深入探讨了ADI Trinamic电机控制器ROS1驱动程序提供的功能，包括：

- 电机运动控制
- 检索电机和控制器信息
- 执行TMC命令
- 获取轴和全局参数值
- 支持多个TMC模块控制设置

所有这些功能都是利用ROS的消息传递系统实现的，使得电机控制器可以轻松集成到基于ROS的系统和应用中。

如需了解更多信息，请访问ADI机器人页面。

探索永不停息

- 查看文章“实现机器人操作系统——ADI Trinamic电机控制器ROS1驱动程序简介”
- 敬请关注未来发表的有关用于ADI Trinamic电机控制器的ROS2的文章！
- 下载Trinamic电机控制器ROS1驱动程序
- 下载Trinamic电机控制器ROS2驱动程序
- 点击此处购买ADI Trinamic电机控制器评估板
- 点击此处购买ADI Trinamic电机

作者简介

Krizelle Paulene Apostol是一名软件系统工程师，在ADI公司菲律宾开发中心的检测、运动和机器人部门工作。她于2019年12月加入ADI公司，工作地点位于菲律宾甲米地。她毕业于菲律宾信心学院，获计算机工程学士学位。她曾参与众多项目，专注于ROS、Gazebo仿真、固件开发、通信协议和算法开发等领域。

作者简介

Jamila "Jam" Aria Macagba是一名高级软件系统工程师，在ADI公司菲律宾开发中心的检测、运动和机器人部门工作。她于2018年7月加入ADI公司，工作地点位于菲律宾甲米地。她毕业于菲律宾大学洛斯巴洛斯分校，获电气工程学士学位。她主要负责ROS系统中的ROS驱动程序开发与集成工作。

作者简介

Maggie Maralit是一名软件系统设计工程经理，在ADI公司菲律宾开发中心的检测、运动和机器人部门工作。她于2019年4月加入ADI公司，工作地点位于菲律宾甲米地。她毕业于菲律宾大学洛斯巴洛斯分校（位于菲律宾拉古纳），获计算机科学学士学位。她目前在菲律宾工厂率领工程师小组，为工业机器人项目提供支持。从2009年至2010年，Maggie在惠普担任应用专家；从2010年至2013年，在Canon Information Technologies Phils., Inc.担任高级软件工程师；从2013年至2015年，在Ionics EMS, Inc.担任固件开发工程师；从2015年至2019年，在新加坡大陆汽车公司担任高级嵌入式软件工程师。