ورقة بحثية مجلة مفتوحة

سلسلة تعلم استخدام نظام ROS2 في نظام التشغيل Windows10 مباشرة (الجزء الثاني)

د. عيسى الغنام* ، د. إياد حاتم**

*(كلية الهندسة، جامعة المنارة ، البريد الإلكتروني: Essa.Alghannam@manara.edu.sy)

**(كلية الهندسة، جامعة المنارة، البريد الإلكتروني: iyadhatem@manara.edu.sy)

الملخص

في هذا الجزء من السلسلة، سنتكلم بداية عن خدمة DDS وهو اختصار لـ Data Distribution Service، ثم سنتعرف على المحاكي Turtlesim وهو محاكي خفيف الوزن لتعلم ROS2. فهو يوضح ما يفعله ROS2 على المستوى الأساسي لإعطائك فكرة عما ستفعله باستخدام روبوت حقيقي أو محاكاة روبوت لاحقًا. وتوفر روبوت على شكل سلحفاة في بيئة افتراضية ثنائية الأبعاد حيث يتم التحكم بحركته من خلال التحكم بسرعته الخطية والزاوية. كما سنتعرف في هذا الجزء على الأداة rqt وهي أداة واجهة مستخدم رسومية (GUI) لـ ROS2. وفي النهاية سنتعرف على مفهوم العقد NODES وآلية التعامل معها.

كلمات مفتاحية – ROS2, DDS, TURTLESIM, RQT, NODES.

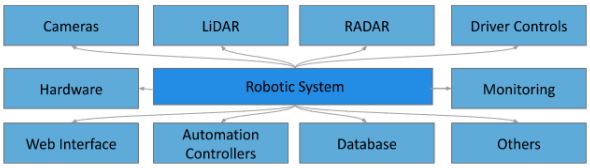
Abstract

In this part of the series, we will first talk about DDS, an abbreviation for Data Distribution Service, then we will learn about Turtlesim, which is a lightweight simulator for learning ROS2. It explains what ROS2 does at a basic level to give you an idea of what you would do with a real robot or a robot simulation later. It provides a turtle-shaped robot in a two-dimensional virtual environment where its movement is controlled by controlling its linear and angular speed. In this part, we will also learn about the *rqt* tool, which is a graphical user interface (GUI) tool for ROS2.

Keywords – ROS2, DDS, TURTLESIM, RQT.

I. استخدام نظام DDS في ROS2 لبناء اتصالات موثوقة في أنظمة الروبوتات

تعد أنظمة الروبوتات الحديثة معقدة بسبب الحاجة إلى دمج أنواع مختلفة من أجهزة الاستشعار والمحركات ومكونات الأجهزة الأخرى من أجل أداء المهام في مجموعة متنوعة من البيئات. تتطلب هذه الأنظمة أيضًا برامج متطورة لمعالجة البيانات من أجهزة الاستشعار، والتحكم في المحركات، واتخاذ القرارات بناءً على تلك البيانات. يعد تعقيد أنظمة الروبوتات الحديثة ضروريًا لتحقيق مستويات عالية من الأداء والوظائف.



الشكل 1 . البنى الأساسية لنظام روبوتي معقد

من المهم أن يكون لدينا نظام قوي وموثوق لمعالجة البيانات الواردة من أجهزة الاستشعار. قد يشمل ذلك خوارزميات perception و localization والتخطيط planning والتحكم بالإضافة إلى برامج التواصل بين المكونات.

تعد خدمة توزيع البيانات (DDS) معيارًا لاتصالات النشر والاشتراك التي تركز على البيانات في الأنظمة الموزعة. وهو مصمم لتمكين الاتصال في الوقت الحقيقي، عالي الإنتاجية، وزمن الوصول المنخفض بين الأجهزة والأنظمة، مع دعم مدمج لجودة الخدمة والأمان والموثوقية. نظام DDS هو نظام مراسلة للنشر والاشتراك يسمح لأجهزة أو عقد متعددة بالتواصل مع بعضها البعض في نظام واحد أو نظام موزع من خلال تبادل البيانات. يختلف هذا النظام عن بنيات الشبكات الأخرى مثل point to point وخادم العميل client–server وقائمة الانتظار queuing بعدد من النقاط تشمل:

التركيز على البيانات: تم تصميم نظام DDS لدعم الاتصالات التي تركز على البيانات، حيث تقوم الأجهزة بتبادل البيانات بدلاً من الرسائل أو الأوامر. يتيح ذلك تدفق البيانات في الوقت الفعلي وتوزيعها.

جودة الخدمة: يتضمن نظام DDS دعمًا لمختلف سياسات جودة الخدمة (QoS) التي تسمح للمستخدمين بضبط سلوك النظام لتلبية احتياجاتهم الخاصة.

الأمان: يتضمن نظام DDS دعمًا مدمجًا للأمان، بما في ذلك التشفير والمصادقة والتحكم في الوصول، للحماية من الوصول غير المصرح به والتلاعب.

II. مقدمة عن المحاكي TURTLESIM

المحاكي Turtlesim هو محاكي خفيف الوزن لتعلم ROS2. فهو يوضح ما يفعله ROS2 على المستوى الأساسي لإعطائك فكرة عما ستفعله باستخدام روبوت حقيقي أو محاكاة روبوت لاحقًا. وهو يوفر روبوت على شكل سلحفاة في بيئة افتراضية ثنائية الأبعاد حيث يتم التحكم بحركته من خلال التحكم بسرعته الخطية والزاوية. يتم من خلاله توفير طريقة لمستخدم ros2 المبتدئ للتعرف على نظام ROS والتفاعل معه. كما يدعم أوامر متعددة تستهدف جوانب مختلفة من النظام وتشغيله. يمكن للمتدرب استخدامه لبدء عقدة node، وتعيين معلمة set a parameter، والاستماع إلى موضوع topic، وغير ذلك الكثير. كما سنتعرف في هذا الجزء على الأداة rqt وهي أداة واجهة مستخدم رسومية (GUI) لـ ROS2. كل شيء يتم إجراؤه في الأداة rqt يمكن القيام به في نافذة الأوامر cmd، ولكنها توفر طريقة بصرية ورسومية أكثر سهولة في الاستخدام للتعامل مع عناصر ROS2. وفي هذا الجزء أيضا سنتعرف إلى مفاهيم ROS2 الأساسية، مثل العقد nodes والموضوعات topics والخدمات services. حيث سيتم شرح كل هذه المفاهيم بالتفصيل في أجزاء لاحقة.

III. التحقق من تثبيت الحزمة:

أثناء تنصيب ros2 كما ورد في الجزء الأول من السلسلة، فإن حزمة turtlesim تكون منصبة مسبقا وللتحقق من ذلك افتح طرفية جديدة وشغل الأمر التالي:

```
ros2 pkg executables turtlesim
```

يجب أن يعرض الأمر أعلاه قائمة بالملفات التنفيذية للمحاكي turtlesim:

```
turtlesim draw_square
turtlesim mimic
turtlesim turtle_teleop_key
turtlesim turtlesim_node
```

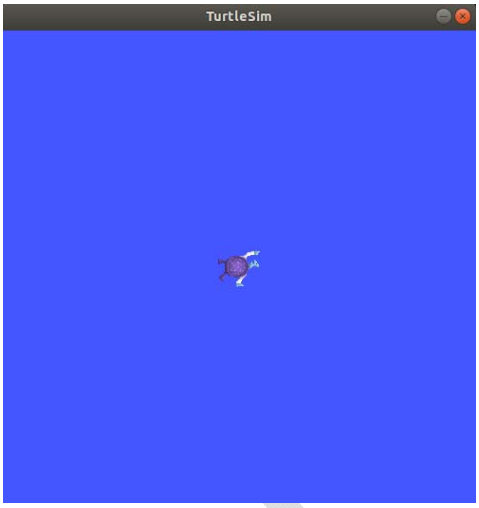
كل ملف تنفيذي من الملفات السابقة هي عبارة عن عقدة node داخل الحزمة package المسماة turtlesim والمنصبة مسبقا مع تنصيب ros2 في windows10. على سبيل المثال turtlesim mimic تزود وظيفة بسيطة لجعل سلحفاة تحاكي سلحفاة أخرى

IV. تشغيل المحاكي TURTLESIM:

لبدء المحاكي Turtlesim، افتح طرفية جديدة مستحضرا نظام ros2 في داخلها كما فعلت ذلك في الجزء الأول ثم أدخل الأمر التالي في هذه الطرفية:

```
ros2 run turtlesim turtlesim_node
```

يجب أن تظهر نافذة المحاكاة، وبها سلحفاة عشوائية في المنتصف.



الشكل 2 . turtlesim_node

في الطرفية، وتحت الأمر الذي أدخلته، سترى رسائل من العقدة node التي قمت بتشغيلها للتو:

```
[INFO] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

هناك يمكنك أن ترى اسم السلحفاة الافتراضي وهو turtlesim والإحداثيات التي تولد فيها، حيث هناك ثلاث احداثيات تمثل موقع الروبوت أو السلحفاة في فضاء ديكارتي ثنائي الابعاد x, y ولها قيم 5.544445cm لكل منهما في الوضع المبين في الشكل أعلاه، وكذلك توجه السلحفاة theta وهي زاوية بين محور الروبوت والمحور الأفقي للبيئة (قيمته صفر في الوضعية المبينة في الشكل أعلاه).

V. استخدام المحاكي TURTLESIM:

افتح محطة طرفية جديدة كما تعلمت سابقا في الجزء الأول مستحضرا ROS2. الآن سنقوم بتشغيل عقدة جديدة للتحكم في السلحفاة التي ولدت في العقدة الأولى في نافذة الأوامر السابقة:

```
ros2 run turtlesim turtle_teleop_key
```

العقدة teleop_key تسمح لك باستخدام أزرار في لوحة المفاتيح لتحريك الروبوت. في هذه المرحلة، يجب أن يكون لدينا ثلاث نوافذ مفتوحة: نافذة تشغيل العقدة turtlesim_node، والتي تحوي معلومات الموقع والتوجيه، ونافذة لتشغيل العقدة التي تخص لوحة المفاتيح turtle_teleop_key ونافذة المحاكي Turtlesimتظهر الروبوت السلحفاة بخلفية زرقاء. قم بترتيب هذه النوافذ بحيث يمكنك رؤية نافذة المحاكي Turtlesim، ولكن أيضًا اجعل المحطة الطرفية التي تعمل فيها turtle_teleop_key نشطة حتى تتمكن من التحكم في السلحفاة في النافذة.

استخدم مفاتيح الأسهم على لوحة المفاتيح للتحكم بالسلحفاة. سوف تتحرك حول الشاشة راسمة المسار الذي اتبعته حتى الآن. حيث يؤدي الضغط على مفتاح السهم إلى تحرك السلحفاة لمسافة قصيرة ثم التوقف.

يمكنك رؤية nodes و topics و services و actions المرتبطة بها، باستخدام الأمر الفرعي list للأوامر المعنية:

```
ros2 node list
ros2 topic list
ros2 service list
ros2 action list
```

سوف نتعلم المزيد عن هذه المفاهيم في أجزاء أخرى من هذه السلسلة. نظرًا لأن الهدف في هذا الجزء هو فقط الحصول على نظرة عامة عن المحاكي Turtlesim، فسوف تستخدم الأداة rqt للاتصال بعض خدمات المحاكي Turtlesimوالتفاعل مع turtlesim_node.

VI. تنصيب واستخدام الأداة rqt:

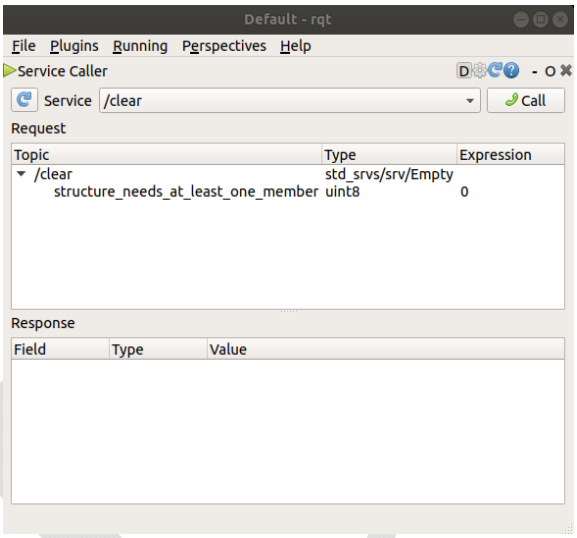
إن الأداة rqt وجميع مكوناتها الإضافية منصبة مسبقا خلال خطوات التنصيب التي مرت معنا في الجزء الأول. افتح طرفية جديدة ونفذ الأمر التالي:

```
rqt
```

عند تشغيل الأداة rqt لأول مرة، ستكون النافذة فارغة؛ ما عليك سوى تحديد Plugins > Services > Service Caller من شريط القائمة في الأعلى.

قد يستغرق الأمر بعض الوقت لـ الأداة rqt لتحديد موقع جميع المكونات الإضافية. إذا قمت بالنقر فوق المكونات الإضافية ولكن بدون رؤية الخدمات أو أي خيارات أخرى، فيجب عليك

إغلاق الأداة rqt وإدخال الأمر rqt --force-discover في الطرفية.



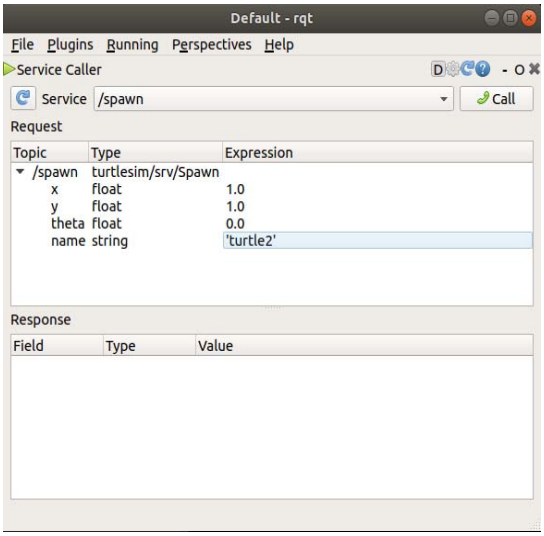
الشكل 3 . الأداة rqt

استخدم زر التحديث الموجود على يسار القائمة المنسدلة Service للتأكد من توفر جميع خدمات عقدة turtlesim الخاصة بك. انقر على القائمة المنسدلة لـ Service لرؤية خدمات turtlesim، وحدد خدمة /spawn.

VII. تجريب خدمة النشر SPAWN SERVICE:

دعنا نستخدم الأداة rqt للاتصال بخدمة /spawn. يمكنك أن تخمن من اسمه أنه ستُكوّن سلحفاة أخرى في النافذة الزرقاء.

امنح السلحفاة الجديدة اسمًا فريدًا، مثل turtle2 عن طريق النقر المزدوج بين علامات الاقتباس المفردة في عمود expression. يمكنك أن ترى أن هذا expression يتوافق مع topic المسمى name وهو من نوع string. بعد ذلك أدخل بعض الإحداثيات الصحيحة لتولد السلحفاة الجديدة فيه، مثل x = 1.0 و y = 1.0.



الشكل 4 . ضبط موقع وتوجيه الروبوت–السلحفاة المولود في البيئة

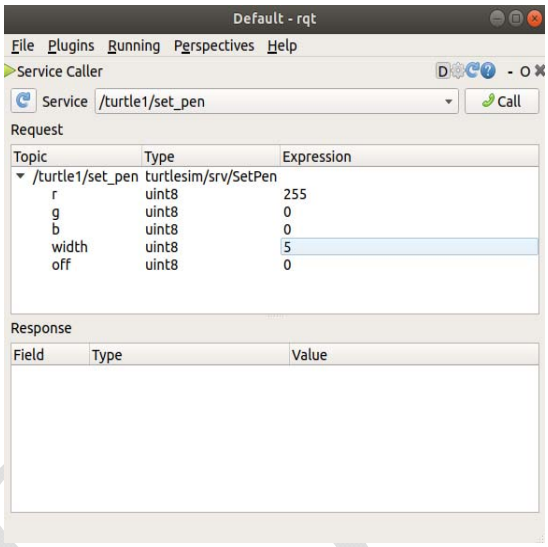
إذا حاولت أن تولد روبوت سلحفاة جديدة بنفس اسم الروبوت السلحفاة الحالية، مثل السلحفاة الافتراضية turtlesim_node 1، فستتلقى رسالة خطأ في الطرفية التي تشغل فيها:

```
[ERROR] [turtlesim]: A turtle named [turtle1] already exists
```

لنشر turtle2، تحتاج بعد ذلك إلى الاتصال بالخدمة بالنقر فوق الزر "call" في الجانب الأيمن العلوي من نافذة الأداة rqt. إذا كانت الخدمة ناجحة، يجب أن ترى سلحفاة جديدة مرة أخرى (بتصميم عشوائي) تُولد عند الإحداثيات التي تدخلها لـ x و y. إذا قمت بتحديث قائمة الخدمات في rqt، فسترى أيضًا أن هناك الآن خدمات متعلقة بالسلحفاة الجديدة، /turtle2/، بالإضافة إلى /turtle1/.

VIII. تجريب خدمة SET_PEN

هنا سنمنح السلحفاة الأولى قلمًا فريدًا باستخدام خدمة:/set_pen



الشكل 5 . ضبط خدمة set_pen لـ turtle1

تحدد قيم r و g و b، التي تتراوح بين 0 و255، لون القلم الذي ترسم به السلحفاة 1، ويحدد العرض سمك الخط.

لجعل السلحفاة 1 ترسم بخط أحمر مميز، قم بتغيير قيمة r إلى 255، وقيمة العرض إلى 5. لا تنس الضغط على CALL الخاص بالخدمة بعد تحديث القيم.

إذا عدت إلى المحطة حيث يتم تشغيل Turtle_teleop_key وضغطت على مفاتيح الأسهم، فسترى أن قلم turtle1 قد تغير. ربما لاحظت أيضًا أنه لا توجد طريقة لتحريك السلحفاة 2. وذلك لأنه لا توجد عقدة Teleop لـ turtle2.

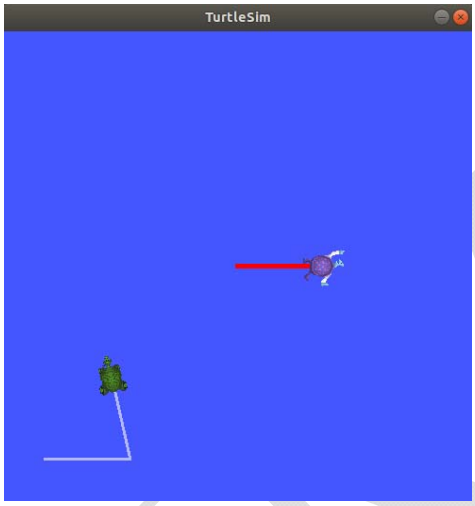
الشكل 6 . الروبوت الثاني يبقى ثابتا لا يتحرك والأول يترك أثرا أحمر على مسار حركته.

نحن بحاجة إلى عقدة Teleop ثانية للتحكم بـ turtle2. ومع ذلك، إذا حاولت تشغيل نفس الأمر كما كان من قبل، ستلاحظ أن هذا الأمر يتحكم أيضًا في turtle1 ولتغيير هذا السلوك ينبغي إعادة تعيين cmd_vel.

في طرفية جديدة نفذ مايلي:

```
ros2 run turtlesim turtle_teleop_key --ros-args --remap
turtle1/cmd_vel:=turtle2/cmd_vel
```

الآن، يمكنك تحريك turtle2 عندما تكون هذه المحطة نشطة، و turtle1 عندما تكون المحطة الطرفية الأخرى التي تقوم بتشغيل Turtle_teleop_key نشطة.



الشكل 7 . التحكم بكلا الروبوتين من خلال لوحة المفاتيح حسب الطرفية المفتوحة

IX. إغلاق TURTLSIM:

لإغلاق بيئة المحاكاة يمكن الضغط على Ctrl + C في الطرفية المفتوحة فيها هذه العقدة turtlesim_node وضغط q في الطرفية الخاصة بعقدة turtle_teleop_key.

X. ROS 2 GRAPH:

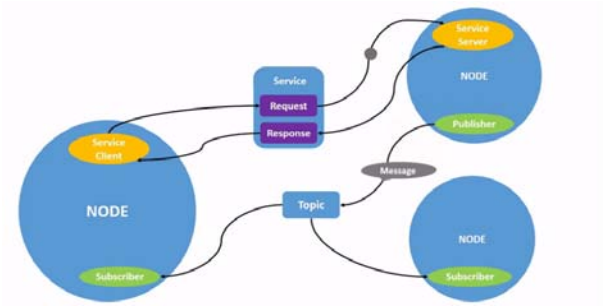
ROS2 عبارة عن برنامج وسيط middleware يعتمد على آلية نشر / اشتراك publish/subscribe mechanism تسمح بتمرير الرسائل message بين عمليات ROS المختلفة. يوجد ROS graph في قلب أي نظام ROS 2، وهو عبارة عن شبكة network من عناصر ROS2 تعالج البيانات معًا في نفس

الوقت. إنه يشمل جميع الملفات التنفيذية executables والاتصالات connections بينها إذا كنت تريد ربطهم map جميعًا وإظهارهم visualize. ويشير ROS graph تحديدا إلى شبكة العقد network of nodes في نظام ROS والوصلات connections التي تتواصل communicate من خلالها. حيث أن:

- العقدة Nodes هي كيان يستخدم ROS للتواصل مع العقد الأخرى.
- الرسائل Messages: نوع بيانات ROS المستخدم عند الاشتراك subscribing أو النشر publishing في موضوع topic.
- المواضيع topic: يمكن للعقد Nodes نشر رسائل publish messages إلى موضوع topic بالإضافة إلى الاشتراك في موضوع subscribe to a topic لتلقي الرسائل receive messages.
- الاكتشاف Discovery: العملية التلقائية التي تحدد العقد من خلالها كيفية التحدث مع بعضها البعض.

XI. العقد Nodes في ROS2

يجب أن تكون كل node عقدة في ROS مسؤولة عن غرض معياري واحد، على سبيل المثال التحكم في محركات العجلات controlling the wheel motors أو نشر بيانات المستشعر من جهاز تحديد المدى بالليزر publishing the sensor data from a laser range-finder. يمكن لكل عقدة node إرسال واستقبال البيانات من العقد الأخرى عبر الموضوعات topics أو الخدمات services أو الإجراءات actions أو المعلمات parameters.



الشكل 8 . تبادل البيانات بين عدة عقد

يتكون النظام الروبوتي الكامل من العديد من العقد التي تعمل في كفرقة موسيقية. في ROS 2، يمكن أن يحتوي ملف تنفيذي واحد (برنامج C ++ ، برنامج Python ، إلخ) على عقد واحد أو أكثر.

يقوم الأمر ros2 pkg executables باستعراض ملفات تنفيذية executable من package حزمة ما.

```
ros2 pkg executables <package_name>
```

على سبيل المثال:

```
ros2 pkg executables turtlesim
```

ينتج:

```
C:\Windows\System32>ros2 pkg executables
turtlesim

turtlesim draw_square.exe
turtlesim mimic.exe
turtlesim turtle_teleop_key.exe
turtlesim turtlesim_node.exe
```

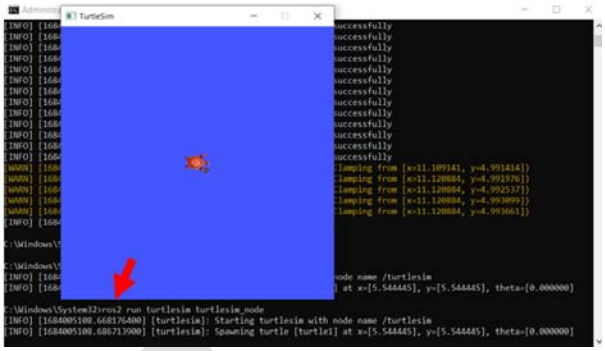
يقوم الأمر ros2 run بتشغيل ملف تنفيذي executable من package حزمة كما يلي.

```
ros2 run <package_name> <executable_name>
```

لتشغيل عقدة ما منها turtlesim، افتح new terminal، وأدخل الأمر التالي:

```
ros2 run turtlesim turtlesim_node
```

سيتم فتح نافذة المحاكي Turtlesim كما في الشكل التالي:



الشكل 9 . تشغيل عقدة turtlesim

هنا، اسم الحزمة هو turtlesim وهي معرفة مسبقا في بيئة ROS2 والاسم القابل للتنفيذ هو turtlesim_node.

إذا كنا لا نعرف اسماء العقد المفتوحة node في جلسة ما. يمكنك العثور عليها باستخدام:

```
ros2 node list
```

مثلا بعد تشغيل العقدة السابقة turtlesim_node اكتب في نافذة جديدة

```
C:\Windows\System32>ros2 node list
/turtlesim
```

جرب الأمر التالي في نافذة جديدة علما أنها عقدة أو ملف تنفيذي من حزمة تسمى turtlesim:

```
ros2 run turtlesim turtle_teleop_key.exe
```

ثم في نافذة جديدة:

```
ros2 node list
```

ينتج:

```
C:\Windows\System32>ros2 node list
/teleop_turtle
/turtlesim
```

جرب الأوامر التالية في نوافذ جديدة:

```
ros2 run turtlesim draw_square.exe
ros2 run turtlesim mimic.exe
```

ثم في نافذة جديدة:

```
ros2 node list
```

ينتج:

```
C:\Windows\System32>ros2 node list
/draw_square
/teleop_turtle
/turtle_mimic
/turtlesim
```

تتيح لك Remapping إعادة تعيين خصائص العقدة الافتراضية، مثل اسم العقدة node name وأسماء الموضوعات topic names وأسماء الخدمات service names وما إلى ذلك، إلى القيم افتراضية custom values. أخيرا، سنستخدم remapping على turtle_teleop_key لتغيير topic موضوع cmd_vel والسلحفاة المستهدفة المسماة turtle2.

الآن، دعنا نعيد تعيين اسم العقدة turtlesim/. في نافذة جديدة، قم بتشغيل الأمر التالي:

```
ros2 run turtlesim turtlesim_node --ros-args --
remap __node:=my_turtle
```

نظرًا لأنك تستخدم ros2 run ، على turtlesim مرة أخرى، سيتم فتح نافذة أخرى من turtlesim. ومع ذلك، إذا عدت الآن إلى النافذة حيث قمت بتشغيل ros2 node list، وقمت بتشغيلها مرة أخرى، فسترى ثلاثة أسماء للعقد:

```
/my_turtle
/turtlesim
/teleop_turtle
```

الآن بعد أن عرفت أسماء العقد الخاصة بك ، يمكنك الوصول إلى مزيد من المعلومات عنها من خلال:

```
ros2 node info <node_name>
```

لفحص أحدث عقدة ، my_turtle، قم بتشغيل الأمر التالي:

```
ros2 node info /my_turtle
```

ros2 node info تعرض معلومات عن قائمة بالمشتركين subscribers والناشرين publishers والخدمات services والإجراءات actions. أي اتصالات الرسم البياني ROS graph connections التي تتفاعل مع تلك العقدة. يجب أن يبدو الإخراج كما يلي:

```
/my_turtle
  Subscribers:
    /parameter_events:
rcl_interfaces/msg/ParameterEvent
    /turtle1/cmd_vel: geometry_msgs/msg/Twist
  Publishers:
    /parameter_events:
rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /turtle1/color_sensor: turtlesim/msg/Color
    /turtle1/pose: turtlesim/msg/Pose
  Service Servers:
    /clear: std_srvs/srv/Empty
    /kill: turtlesim/srv/Kill
    /my_turtle/describe_parameters:
rcl_interfaces/srv/DescribeParameters
```

```
    /my_turtle/get_parameter_types:
rcl_interfaces/srv/GetParameterTypes
    /my_turtle/get_parameters:
rcl_interfaces/srv/GetParameters
    /my_turtle/list_parameters:
rcl_interfaces/srv/ListParameters
    /my_turtle/set_parameters:
rcl_interfaces/srv/SetParameters
    /my_turtle/set_parameters_atomically:
rcl_interfaces/srv/SetParametersAtomically
    /reset: std_srvs/srv/Empty
    /spawn: turtlesim/srv/Spawn
    /turtle1/set_pen: turtlesim/srv/SetPen
    /turtle1/teleport_absolute:
turtlesim/srv/TeleportAbsolute
    /turtle1/teleport_relative:
turtlesim/srv/TeleportRelative
  Service Clients:

  Action Servers:
    /turtle1/rotate_absolute:
turtlesim/action/RotateAbsolute
  Action Clients:
```

حاول الآن تشغيل نفس الأمر على عقدة / teleop_turtle، وشاهد كيف تختلف اتصالاتها عن عقدة my_turtle.

سوف سنتعلم المزيد حول مفاهيم ROS graph connection بما في ذلك أنواع الرسائل message types في الأجزاء القادمة.

XII. خلاصة

يعد المحاكي turtlesim و الأداة rqt وسيلتين جيدتين لفهم نواة عمل ros2. العقدة هي عنصر أساسي من عناصر ROS 2 يخدم غرضًا معياريًا واحدًا في نظام الروبوتات.

في هذا الجزء من السلسلة، استخدمت العقدة التي تم إنشاؤها في package حزمة turtlesim عن طريق تشغيل الملفين التنفيذيين turtlesim_node و turtle_teleop_key.

لقد تعلمت كيفية استخدام ros2 node list لاكتشاف أسماء العقد النشطة و ros2 node info لاستكشاف عقدة واحدة. هذه الأدوات ضرورية لفهم تدفق البيانات في نظام روبوت معقد في العالم الحقيقي.

العقدة هي أحد المشاركين في ROS graph. تستخدم عقد ROS مكتبة عميل ROS للتواصل مع العقد الأخرى. يمكن للعقد نشر المواضيع أو الاشتراك فيها publish or subscribe to Topics.

يمكن للعقد أيضًا توفير الخدمات والإجراءات Services and Actions أو استخدامها. هناك معلمات قابلة للضبط configurable Parameters مرتبطة بالعقدة. يتم إنشاء الاتصالات بين العقد من خلال عملية الاكتشاف الموزعة distributed discovery process. قد توجد العقد في نفس العملية same process أو في عمليات مختلفة different processes أو على أجهزة مختلفة different machines. سيتم وصف هذه المفاهيم بمزيد من التفصيل في الأجزاء التالية.

المراجع:

[1]. https://www.ros.org/
[2]. http://wiki.ros.org/
[3]. http://wiki.ros.org/ROS/Tutorials