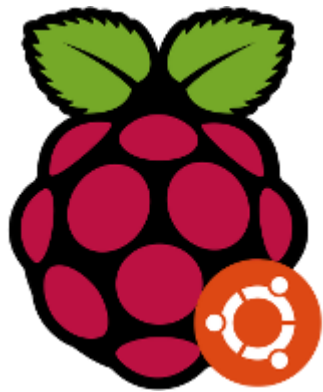




Mechatronics System

Lecture11: Introduction to ROS and RaspberryPI



Prepared By

Eng. Dalia Kass Hanna

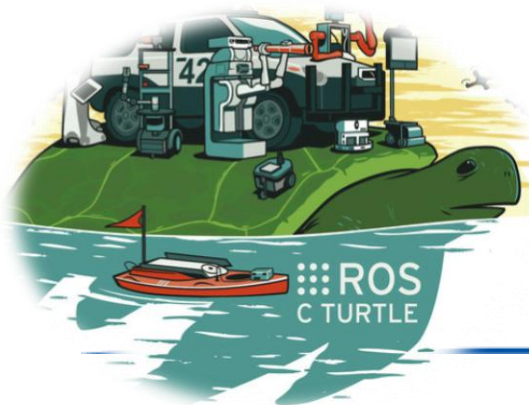
Eng. Aula Jazmati

لمحة عن نظام ROS

هو منصة برمجية تستخدم لتطوير الروبوتات المختلفة بتعديل بسيط على المقاطع البرمجية بدلاً من كتابة برنامج جديد، وهو نظام مفتوح المصدر لذلك يمكن الاستفادة من تطوير بعض الأشخاص للخوارزميات.

ROS مختلف عن الأنظمة الأخرى مثل الـ windows لأنه يحتاج إلى نظام تشغيل آخر ليعمل من خلاله مثل Ubuntu أو Raspbian أو أي نظام تشغيل يقدم الدعم له. لذا نطلق على ROS meta-operating-system أي نظام تشغيل يعمل بوجود نظام آخر.

نظام ROS يقوم بتنظيم تبادل الرسائل والعمليات بين العقد بشكل مشابه تماماً لوظائف أنظمة التشغيل يمتلك هذا النظام الكثير من الإصدارات وتكون الإصدارات مرتبة حسب الحروف.



إصدارات ROS

ROS Diamondback

March 2, 2011



--

ROS C Turtle

August 2, 2010



--

ROS Box Turtle

March 2, 2010



Box Turtle



--

إصدارات ROS

ROS Hydro Medusa

September 4th, 2013



ROS Groovy Galapagos

December 31, 2012



ROS Fuerte Turtle

April 23, 2012



ROS Electric Emys

August 30, 2011



May, 2015



July, 2014



--



--

إصدارات ROS

Distro	Release date	Poster	Tuturtle, turtle in tutorial	EOL date
ROS Melodic Morenia (Recommended)	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)

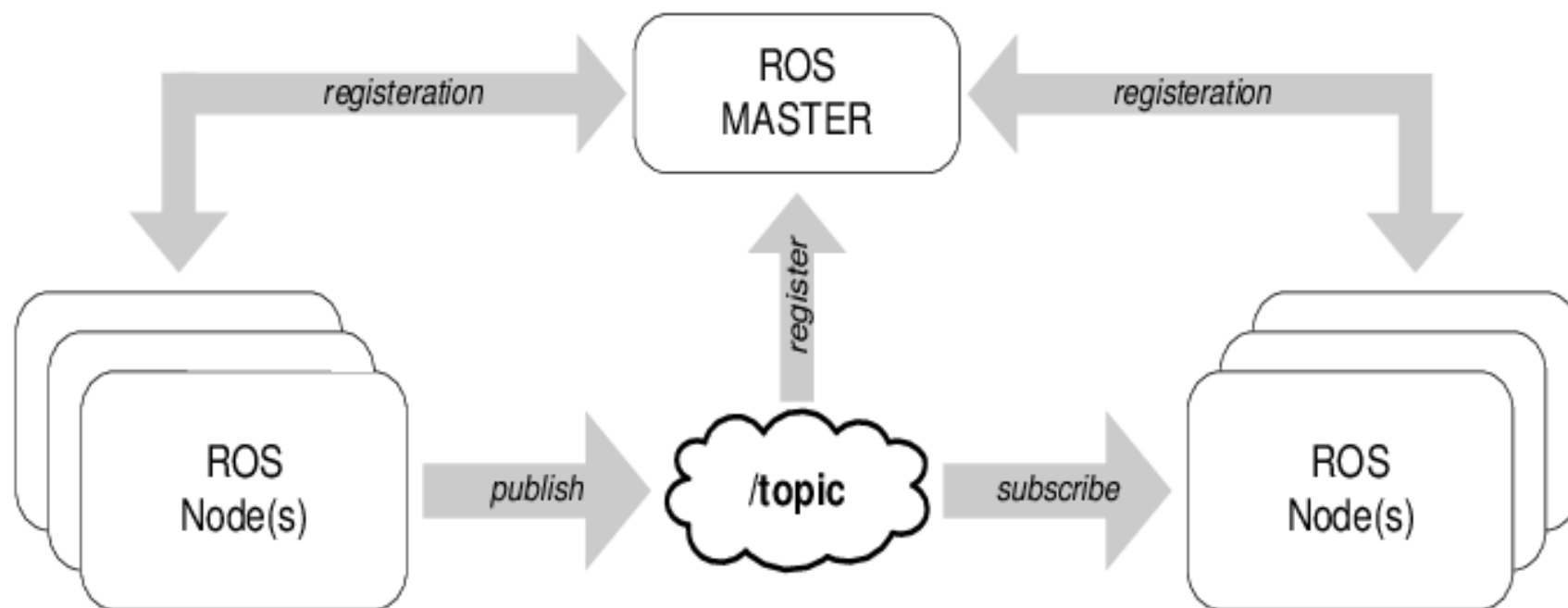
كيف يعمل نظام الـ ROS؟

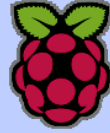
يبدأ النظام بالعمل بتشغيل عقدة رئيسية ROS Master هذه العقدة الرئيسية تسمح لباقي العقد بالتواصل مع بعضها البعض.

إن كل مهمة أساسية تعمل في ROS تُعدّ عقد (nodes) وتتواصل مع بعضها بواسطة رسائل (message) من المعلومات وذلك ضمن مواضيع (topics) وهو المكان الذي يتم نشر رسائل المعلومات فيه وتقوم أداة تدعى bag files بتسجيل هذه الرسائل .

إن الميزة في استخدام العقد أنه في حال توقفت أحد العقد لا يؤثر هذا على عمل بقية النظام ، كما أنه يمكن برمجة هذه العقد بأي لغة يدعمها النظام C++,C,Python,Java.

كيف يعمل نظام الـ ROS؟

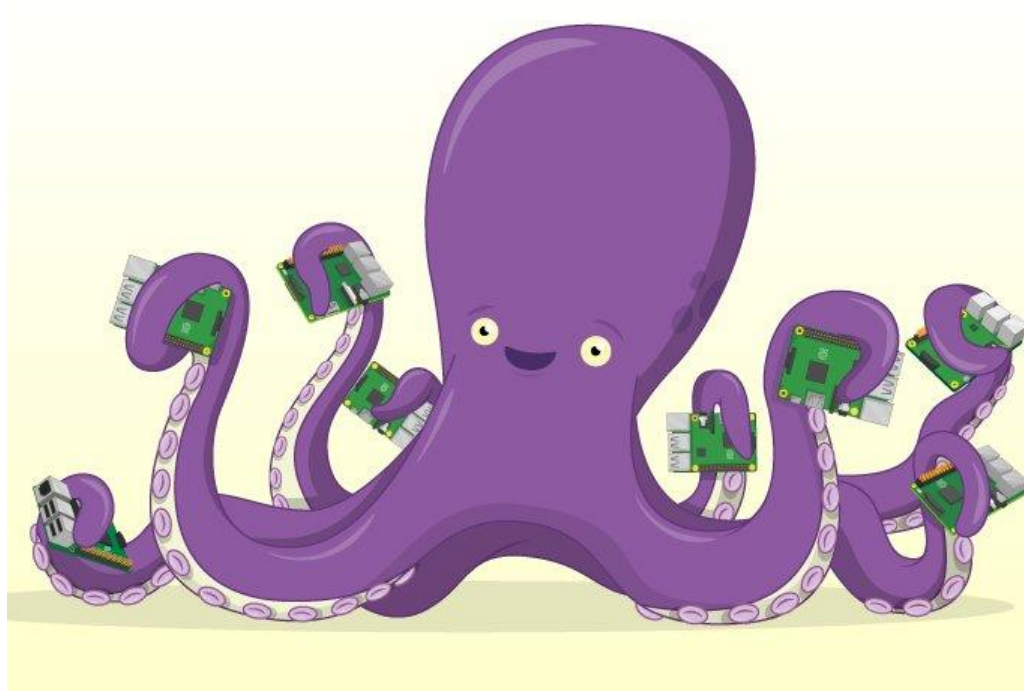




التطبيق العملي:

مراحل العمل ضمن نظام ROS باستخدام Raspberry Pi 3 Model B

أثناء التشغيل online يجب فتح نافذة سطر أوامر جديدة كل مرة وذلك مع الأوامر التي تعمل



١- إغلاق وإيقاف كل العقد المنشأة سابقاً للبدء من جديد:

roscore

للبدء في استخدام ROS علينا بداية بتشغيل عقدة رئيسية، يتم ذلك بكتابة التعليمة roscore

ملاحظة:

Kill -9 rosmaster

لإيقاف ROS

roscore Kill -a

لإيقاف العقد المنشأة سابقاً

-2 تشغيل Nodes:

In a new terminal

```
roslaunch turtlesim turtlesim_node
```

In a new terminal

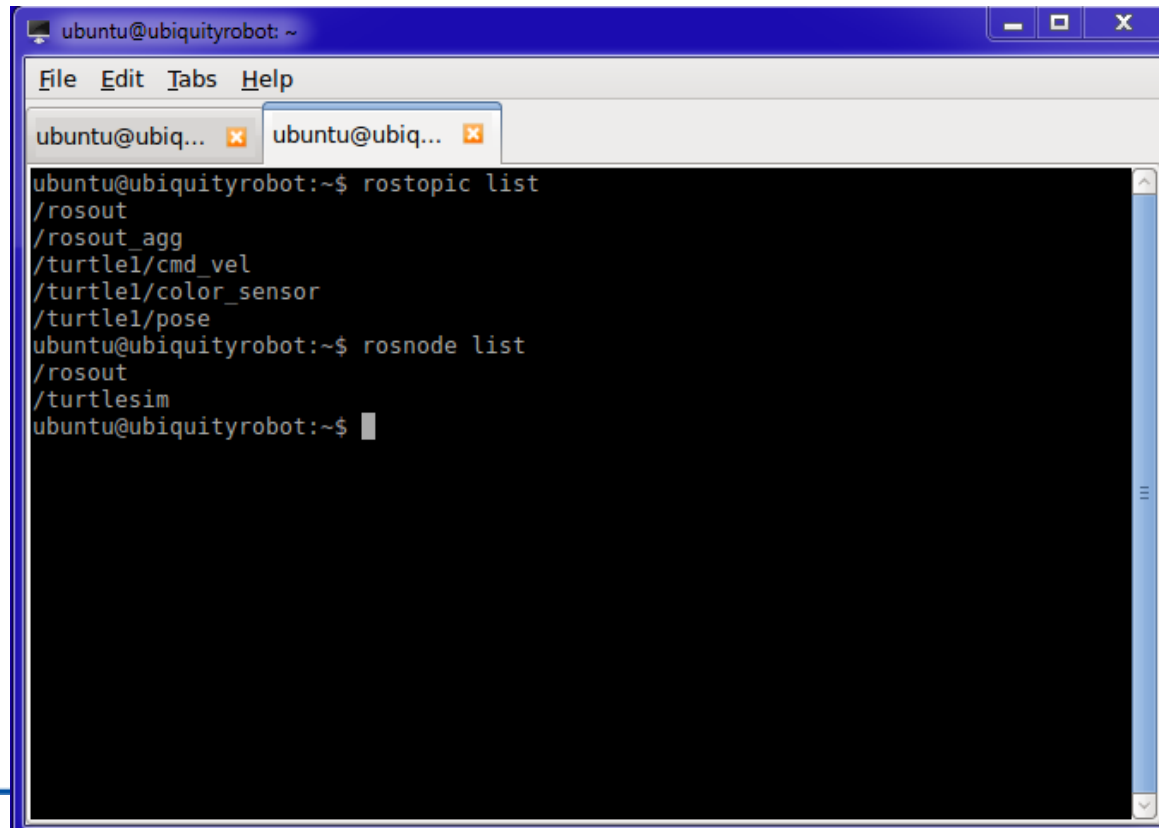
```
roslaunch turtlesim turtle_teleop_key
```



٣- المواضيع rostopic :

تعطي معلومات عن مواضيع ROS بما في ذلك العقد التي تقوم بنشر المعلومات publishers،
العقد التي تقوم بمتابعة المعلومات subscribers، معدل نشر الرسائل ورسائل ROS

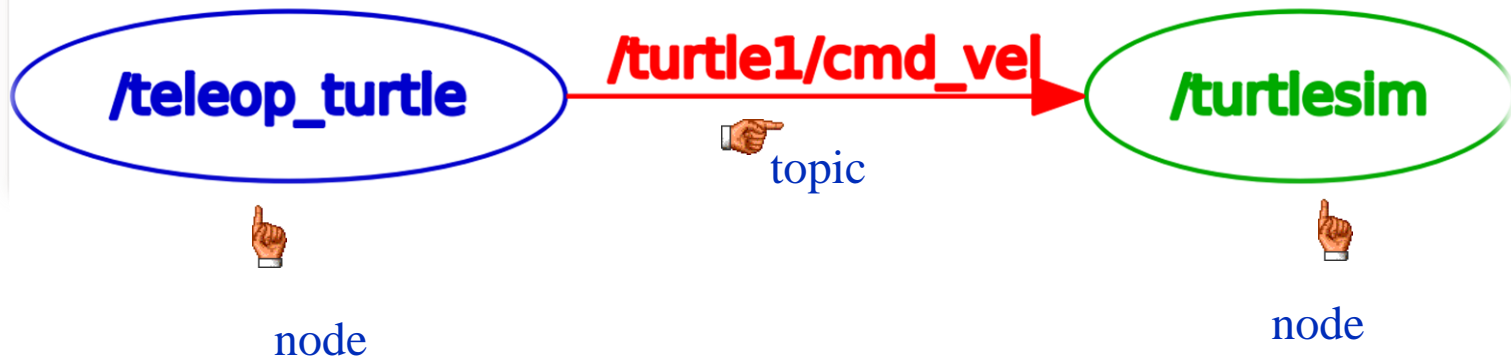
rostopic list
roswode list

A terminal window titled 'ubuntu@ubiquityrobot: ~' with a menu bar (File, Edit, Tabs, Help) and two tabs. The terminal shows the execution of 'rostopic list' and 'roswode list' commands. The output of 'rostopic list' includes '/rosout', '/rosout_agg', '/turtle1/cmd_vel', '/turtle1/color_sensor', and '/turtle1/pose'. The output of 'roswode list' includes '/rosout' and '/turtlesim'.

```
ubuntu@ubiquityrobot:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
ubuntu@ubiquityrobot:~$ roswode list
/rosout
/turtlesim
ubuntu@ubiquityrobot:~$
```

٤- رسم المخطط التمثيلي للنظام

roslaunch rqt_graph rqt_graph



٥- عرض سلسلة المواضيع

rostopic -h

```
^Cubuntu@ubiquityrobot:~$ rostopic -h
rostopic is a command-line tool for printing information about ROS Topics.
```

Commands:

rostopic bw	display bandwidth used by topic
rostopic delay	display delay of topic from timestamp in header
rostopic echo	print messages to screen
rostopic find	find topics by type
rostopic hz	display publishing rate of topic
rostopic info	print information about active topic
rostopic list	list active topics
rostopic pub	publish data to topic
rostopic type	print topic or field type

Type `rostopic <command> -h` for more detailed usage, e.g. `'rostopic echo -h'`

```
ubuntu@ubiquityrobot:~$ rostopic echo
Usage: rostopic echo [options] /topic
```

```
rostopic: error: topic must be specified
ubuntu@ubiquityrobot:~$ rostopic echo [topic]
WARNING: topic [/topic] does not appear to be published yet
```

rostopic -h

rostopic bw display bandwidth used by topic
rostopic echo print messages to screen
rostopic hz display publishing rate of topic
rostopic list print information about active topics
rostopic pub publish data to topic
rostopic type print topic type

rostopic

bw echo find hz info list pub type

rostopic list -v

Published topics:

- * /turtle1/color_sensor [turtlesim/Color] 1 publisher
- * /turtle1/cmd_vel [geometry_msgs/Twist] 1 publisher
- * /rosout [roscpp_msgs/Log] 2 publishers
- * /rosout_agg [roscpp_msgs/Log] 1 publisher
- * /turtle1/pose [turtlesim/Pose] 1 publisher

Subscribed topics:

- * /turtle1/cmd_vel [geometry_msgs/Twist] 1 subscriber
 - * /rosout [roscpp_msgs/Log] 1 subscriber
-

عرض نوع بيانات الرسالة التي ستنتشر ضمن الموضوع

rostopic type /turtle1/cmd_vel

 Topic name

عرض تفاصيل الرسالة

rosmmsg show geometry_msgs/Twist

 Message type

```
ubuntu@ubiquityrobot:~$ rosmmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

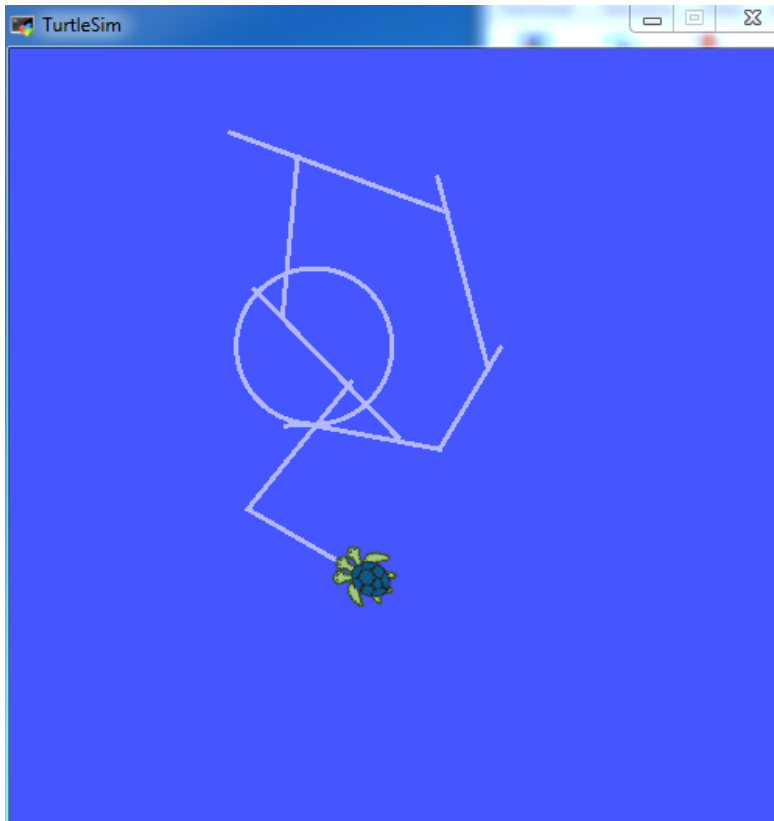
عرض البيانات التي ستنتشر ضمن الموضوع

rostopic echo /turtle1/cmd_vel



**Topic
name**



```
Last login: Fri Nov  2 23:33:48 2018 from 10.42.0.146
ubuntu@ubiquityrobot:~$ rostopic echo /turtle1/cmd_vel
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 2.0
```



rostopic echo /turtle1/pose

```
ubuntu@ubiquityrobot:~$ rostopic echo /turtle1/pose
x: 4.39143180847
y: 5.73268318176
theta: -3.08820581436
linear_velocity: 0.0
angular_velocity: 0.0
---
x: 4.39143180847
y: 5.73268318176
theta: -3.08820581436
linear_velocity: 0.0
angular_velocity: 0.0
```

نشر الرسالة إلى الموضوع

 **Topic name**  **Message type**

```
rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist --  
'[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
```

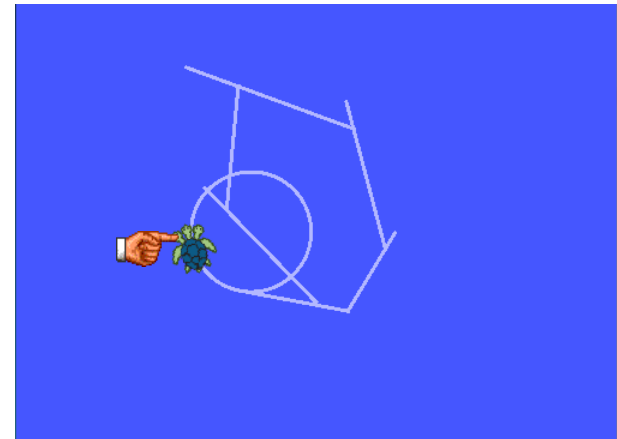
x,y,z
Linear velocity

x,y,z
Angular velocity

The turtle sim should now show a turtle going around in an arc for a time.

Break down of previous command:

- rostopic pub : published the information
- -1 : has the publisher send one command then exit
- /turtle1/cmd_vel : name of topic to publish to
- geometry_msgs/Twist : message type to use
- -- : lets parser know following informations are not an option (important for negative numbers)
- '[x, y, z] (2x)' : YAML velocity vectores. For more on YAML syntax, please see this [link](#).



- This command will publish messages to a given topic:

```
rostopic pub
```

- This option (dash-one) causes rostopic to only publish one message then exit:

```
-1
```

- This is the name of the topic to publish to:

```
/turtle1/cmd_vel
```

- This is the message type to use when publishing to the topic:

```
geometry_msgs/Twist
```

- This option (double-dash) tells the option parser that none of the following arguments is an option. This is required in cases where your arguments have a leading dash -, like negative numbers.

```
--
```

- As noted before, a geometry_msgs/Twist msg has two vectors of three floating point elements each: `linear` and `angular`. In this case, `'[2.0, 0.0, 0.0]'` becomes the linear value with `x=2.0`, `y=0.0`, and `z=0.0`, and `'[0.0, 0.0, 1.8]'` is the angular value with `x=0.0`, `y=0.0`, and `z=1.8`. These arguments are actually in YAML syntax, which is described more in the [YAML command line documentation](#).

```
'[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
```

نشر الرسالة إلى الموضوع بشكل مستمر

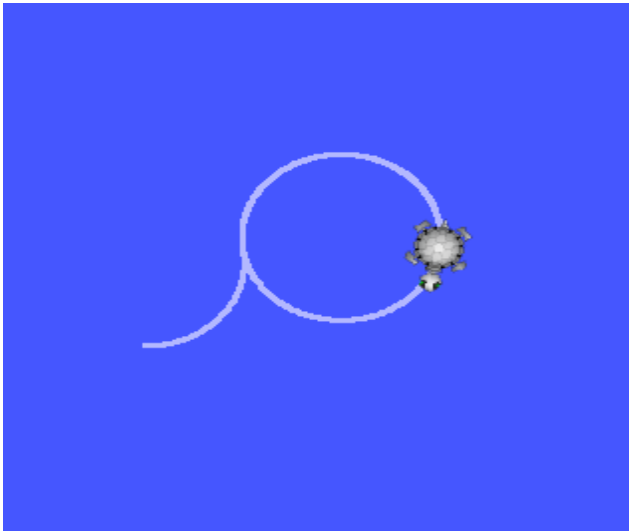
```
rostopic pub /turtle1/cmd_vel geometry_msgs/Twist  
-r 1 -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
```

Topic name

Message type

x,y,z
Linear velocity

x,y,z
Angular velocity

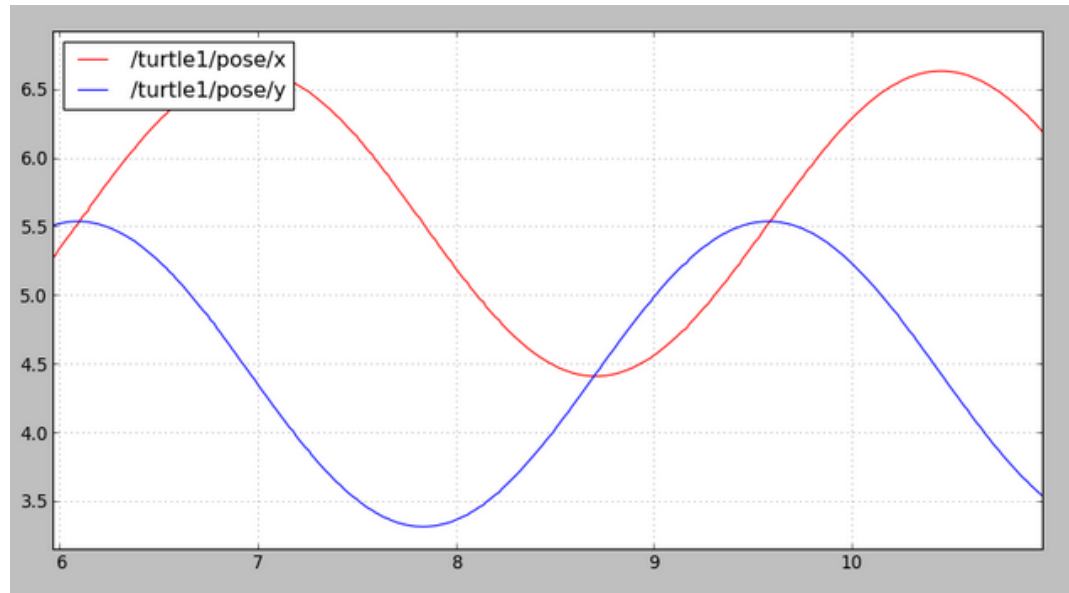


rostopic hz /turtle1/pose

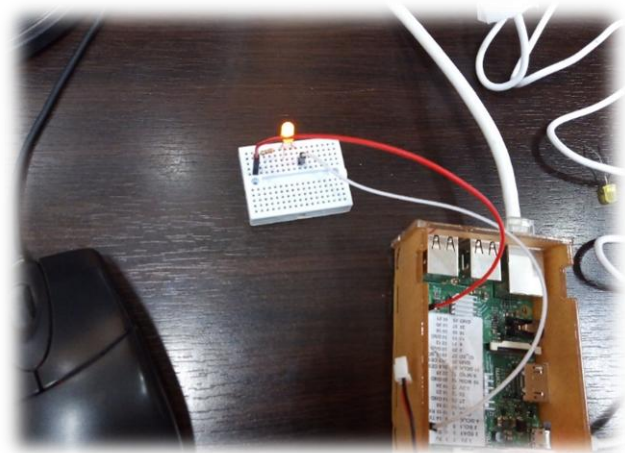
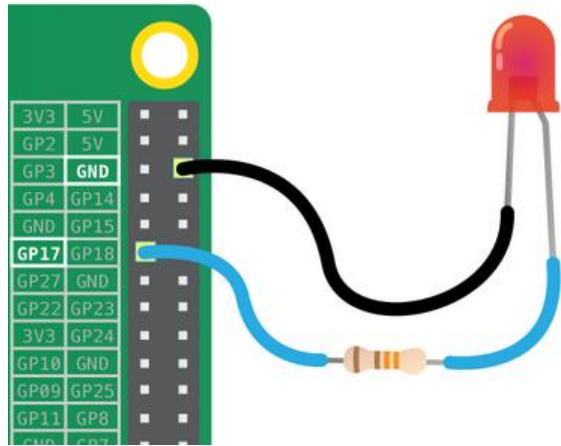
عرض سرعة نشر البيانات إلى الموضوع

roslaunch rqt_plot rqt_plot

عرض مخطط البيانات التي تم إرسالها إلى الموضوع



التطبيق العملي



led.py - /home/ubuntu/catkin_ws/src/led.py (3.5.2)

File Edit Format Run Options Window Help

```
import rospy
from std_msgs.msg import String
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
led = 11
Frequency = 50
Stop = 0
GPIO.setup(led, GPIO.OUT)
pwm1 = GPIO.PWM(led, Frequency)
pwm1.start(Stop)
def StopLED():
    pwm1.ChangeDutyCycle(Stop)
def show():
    pwm1.ChangeDutyCycle(40)
    time.sleep(2)
    pwm1.ChangeDutyCycle(70)
    time.sleep(2)
    pwm1.ChangeDutyCycle(95)
```

إنشاء ملف ضمن IDLE أو من سطر الأوامر وتخزينه
ضمن المسار المبين

استدعاء الموديولات اللازمة

تهيئة الترقيم حسب الطريقة
الفيزيائية وحجب رسائل التحذير
وتهيئة القطب ١١ كخرج

إنشاء عنصر pwm مرتبط بالقطب 11 وبتردد 50 وقيمة
Duty cycle ابتدائية 0

إنشاء تابع لإيقاف تشغيل pwm

إنشاء تابع لتشغيل pwm
ب Duty cycle متعددة

```
def CommandCallback(commandMessage):
    command = commandMessage.data
    if command == 'show':
        print('ChangeDutyCycle')
        show()
    else:
        print('Unknown command, stopping instead')
        StopLED()
```

```
rospy.init_node('driver')
rospy.Subscriber('command', String, CommandCallback)
rospy.spin()
print('Shutting down')
StopLED()
GPIO.cleanup()
```

إنشاء تابع للرسائل بمتغير (String message)
حيث يستدعي الرسالة التي
نكتبها من سطر الأوامر (ننشرها عبر الموضوع)
وبحسب محتوى الرسالة ينفذ أوامر (استدعاء أحد
التوابع في مثالنا)

إنشاء عقدة باستخدام مكتبة Ros
ضمن لغة python باسم driver
للتواصل مع ROS master

الاشتراك (subscribe) في موضوع
command بمتحول من نوع
string والتابع المنشأ للرسائل

استدعاء طريقة spin التي
تنتظر وصول الرسائل وتنفذ
التعليمات التالية للخروج عند
الضغط على CTRL+C

التطبيق العملي

- ننتقل إلى سطر الأوامر ثم إلى المسار الذي تم تخزين المثال السابق عليه
- نغير من **permissions** سماحية الملف «**chmod** “**change mode**”»

Read – Allowed to read files

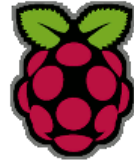
Write – Allowed to write/modify files

eXecute – Read/write/delete/modify/directory

- نقوم بتنفيذه ضمن python
- عند التأكد من عدم وجود أخطاء نقوم بنشر رسائل عبر الموضوع ونلاحظ النتائج !

```
ubuntu@ubiquityrobot:~$ cd catkin_ws
ubuntu@ubiquityrobot:~/catkin_ws$ cd src
ubuntu@ubiquityrobot:~/catkin_ws/src$ chmod +x led.py
ubuntu@ubiquityrobot:~/catkin_ws/src$ python led.py
```

```
ubuntu@ubiquityrobot:~/catkin_ws/src$ rostopic pub -1 /command std_msgs/String "show"
publishing and latching message for 3.0 seconds
ubuntu@ubiquityrobot:~/catkin_ws/src$ rostopic pub -1 /command std_msgs/String "stop"
publishing and latching message for 3.0 seconds
ubuntu@ubiquityrobot:~/catkin_ws/src$
```



Thank You