sudo apt-get install ros-melodic-rosserial

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

sudo usermod -a -G dialout kati

sudo chmod a+rw /dev/ttyUSB0

$ ls -l /dev/ttyACM\*

$ sudo chmod a+rw /dev/ttyACM0

$ udevadm info --name=/dev/ttyACM0 --attribute-walk

SUBSYSTEMS=="usb", ATTRS{manufacturer}=="Roboteq", ATTRS{product}=="Motor Controller", SYMLINK+="agv/roboteq", ENV{ID\_MM\_DEVICE\_IGNORE}="1", MODE="0666"

$ sudo cp myrules.rules /etc/udev/rules.d/

………………………………………………………………………………………..

New Terminal

$ cd ~/Arduino/libraries

$ rm -rf ros\_lib

$ rosrun rosserial\_arduino make\_libraries.py .

$ rosrun rosserial\_client make\_libraries ~/Arduino/libraries

เปิด terminal และ roscore

$ roscore

Check Device

$ ls /dev/tty\*

เปิด terminal และ rosrun

$ rosrun rosserial\_python serial\_node.py /dev/ttyUSB0

เปิด terminal และดู topic

$ rostopic list

$

<launch>

<node name="Speaker" pkg="test" type="basic\_pub.py" />

<node name="Listener" pkg="test" type="basic\_sub.py" />

</launch>

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22. | #include "ros.h"  #include "std\_msgs/String.h"  ros::NodeHandle nh;    std\_msgs::String str\_msg;  ros::Publisher chatter("chatter", &str\_msg);    char hello[13] = "hello world!";    void setup()  {  nh.initNode();  nh.advertise(chatter);  }    void loop()  {  str\_msg.data = hello;  chatter.publish(&str\_msg);  nh.spinOnce();  delay(1000);  } |

บรรทัดที่ 1 เรียกใช้งาน ros.h c ต้องทำก่อนเสมอ

บรรทัดที่ 2 เรียกใช้งาน std\_msgs/String.h

บรรทัดที่ 3 สร้างตัวแปร nh จากฟังก์ชัน ros::NodeHandle ตัว Node handle ขึ้นมา หน้าที่ของตัวนี้คือการทำให้โปรแกรมของเราสามารถสร้าง publishers และ subscribers ได้ อีกทั้งมันยังเป็นตัวที่ช่วยจัดการดูแล การเชื่อมต่อผ่าน Serial port

บรรทัดที่ 5 สร้างตัวแปร str\_msg จากฟังก์ชัน std\_msgs::String

บรรทัดที่ 6 สร้างตัวแปร chatter จากฟังก์ชัน ros::Publisher ซึ่งก็คือ publisher ที่มี topic ชื่อว่า “chatter” ส่วน parameter ตัวที่สอง คือการบอกถึง message ที่จะใช้ publish ออกไป

บรรทัดที่ 8 สร้างตัวแปร hello เป็นตัวแปรประเภท char เก็บข้อความ "hello world!"

บรรทัดที่ 12 สั่งเริ่มต้นทำงาน nh โดยการสั่ง initialize

บรรทัดที่ 13 สั่ง advertise เพื่อที่จะบอกว่ามี topic ไหนบ้างที่เราจะ publish หรือ subscribe ในที่นี้เราได้ประกาศว่าจะ publish ไปที่ topic chatter

บรรทัดที่ 18 นำตัวแปร hello ไปเก็บไว้ที่ str\_msg.data

บรรทัดที่ 19 publish “Hello World” ไปที่ topic “chatter”

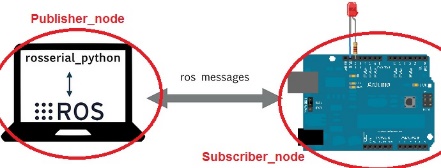
บรรทัดที่ 20 เรียก ros::SpinOnce() คือการสั่งการทำงานทุกตัวใน

บรรทัดที่ 21 หน่วงเวลา 1000 ms

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20. | #include <ros.h>  #include <std\_msgs/Empty.h>  ros::NodeHandle nh;  void messageCb( const std\_msgs::Empty& toggle\_msg){  digitalWrite(13, HIGH-digitalRead(13));  }  ros::Subscriber<std\_msgs::Empty> sub(“toggle\_led”, &messageCb );  void setup()  {  pinMode(13, OUTPUT);  nh.initNode();  nh.subscribe(sub);  }  void loop()  {  nh.spinOnce();  delay(1);  } |

บรรทัดที่ 1 เรียกใช้งาน ros.h c ต้องทำก่อนเสมอ

บรรทัดที่ 2 เรียกใช้งาน std\_msgs/ Empty.h เป็นไลบรารีการส่งข้อความที่จัดการการสื่อสารระหว่างผู้ publishers และ subscribers เนื่องจากแอปพลิเคชันนี้ไม่ต้องการแลกเปลี่ยนข้อมูลจริงๆ จึงมีการใช้ข้อความว่างเปล่า empty และจะมีแค่ subscriber เท่านั้นโดยไม่มีการสร้าง publisher



บรรทัดที่ 4 สร้างตัวแปร nh จากฟังก์ชัน ros::NodeHandle

บรรทัดที่ 6-8 กำหนดวิธีการเมื่อข้อความใน Topic ซึ่งในที่นี้ค่าในพิน 13 จะถูกสลับ

บรรทัดที่ 10 สร้าง Subscriber ที่ Topic “ toggle\_led” ใน message “messageCb”

บรรทัดที่ 14 กำหนดพิน 13 เป็น output

บรรทัดที่ 15 สั่งเริ่มต้นทำงาน nh โดยการสั่ง initialize

บรรทัดที่ 16 สั่ง subscribe เพื่อเริ่มต้นรับฟัง publisher

บรรทัดที่ 20 เรียก ros::SpinOnce() คือการสั่งการทำงานทุกตัวใน

บรรทัดที่ 21 หน่วงเวลา 1000 ms

เปิด terminal และดู topic

rosrun rosserial\_python serial\_node.py /dev/ttyUSB0

$ rostopic list

$ rostopic pub toggle\_led std\_msgs/Empty --once