JURNAL ROS

22 MEI 2022

COURSE : ROS BASICS

ROS BASIC | CATKIN WORKSPACE DAN PACKAGE

CATKIN WORKSPACE

Sebelum kita membuat sebuah package, hal pertama yang harus kita harus membuat area kerja kita atau workspace. Idealnya kita harus membuat satu workspace untuk satu project, jadi misal saat ini kita membuat misi sauvc , maka kita akan membuat satu workspace catkin yang dimana isinya merupakan kumpulan kumpulan package yang akan dipakai untuk misi sauvc, perlu di ketahui juga bahwa antara workspace bisa terjadi komunikasi juga, tapi demi kenyamanan dan kerapihan, di sarankan kalian untuk memiliki sistem satu workspace per project.

Nah sekarang kita bahas, apa si workspace itu ? dasarnya workspace catkin sesuai namanya, merupakan tempat bekerja package kalian, dasarnya sebuah package ros pasti akan berada pada sebuah workspace yang dimana package kalian akan di compile atau di build berdasarkan preferensi kalian, intinya **package tidak akan bisa terbangun apabila tidak berada di dalam workspace**

Nah aku akan menjelaskan cara membuat workspace catkin kalian sebelum kalian akan membuat package ros.

*pastikan kalian sudah menginstall ros , aku gak akan ngecover masalah penginstalan pada dokumentasi ini, pada halaman wiki ros sudah dicover dengan cukup jelas dan mudah

Langkah Langkah pembuatan workspace catkin

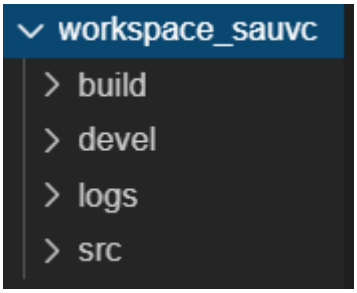
1. Buka terminal kalian dan jalankan command `roscore` apabila ros sudah berjalan dengan benar maka buka tab terminal baru

2. Setelah tab terminal baru sudah terbuka, pergilah ke directory yang kalian inginkan, baca dokumentasi dasar dasar linux ku kalau kalian masih ada kebingungan. Pada tutorial ini aku akan menggunakan directory `home/abin/banyubramanta`

3. Setelah masuk ke dalam directory tersebut, buatlah folder baru menggunakan command `mkdir nama_folder` , pada tutorial ini nama folder nya adalah `workspace_sauvc`

4. Setelah masuk ke dalam directory tersebut, buatlah folder khusus tempat package kalian akan disimpan, folder **harus bernama** “`src`”, kalau engga, workspace tidak dapat terbangun.

5. Setelah itu kalian bisa menggunakan command `catkin build` untuk meng-inisiasi pembuatan workspace kalian, dan setelah build success kalian cek ada folder apa di workspace kalian, kalau sudah sesuai dengan gambar dibawah, berarti kalian telah berhasil membuat workspace kalian :D



Ros merupakan framework yang digunakan untuk mengatur program robot di background. Seluruh program file yang dijalankan untuk satu fungsi tertentu di jalankan di dalam satu package, apa itu package ?

Package merupakan sebuah paket kesatuan yang mengatur program ros, ada beberapa komponen dalam sebuah package, diantaranya

- **Launch folder**
  launch folder berisi file launch yang dapat digunakan untuk memulai node dalam ros

- **Src folder**
  src folder berisi berbagai program yang akan dijalankan menggunakan program ros

- **Cmakelist.txt**
  merupakan file txt yang digunakan untuk mengcompile di akhir saat ingin menambahkan update pada package ros kita

- **Package.xml**
  merupakan file yang dapat diubah untuk merubah dependency ros package

Launch file

Launch file , dapat dibuat dengan menggunakan perintah sederhana seperti *touch* pada terminal, ekstensi file ini merupakan **.launch** , dan jangan lupa untuk menggunakan chmod untuk meng-enable agar file ini dapat di execute oleh ros.

Contoh isi launch file:

```
    <launch>
  <!-- turtlebot_teleop_key already has its own built in velocity smoother -->
  <node pkg="turtlebot_teleop" type="turtlebot_teleop_key.py" name="turtlebot_teleop_keyboard"  output="screen">
    <param name="scale_linear" value="0.5" type="double"/>
    <param name="scale_angular" value="1.5" type="double"/>
    <remap from="turtlebot_teleop_keyboard/cmd_vel" to="/cmd_vel"/>   <!-- cmd_vel_mux/input/teleop"/-->
  </node>
</launch>
```

fungsi dari tag node adalah untuk mengarahkan value2 dari launch file ini, anggep arah program mau kemana, tapi bukan programnya ngapain

pkg : package name

type : nama dan type file, lebih ke nama sih

nama : nama node yang nanti di set di file python nya

output: channel output dari launch file ini (masi aga rancu ini soalnya di the construct masih menggunakan screen)

src

source, atau program dari sebuah package yang nanti akan di execute oleh launch file, bisa dibilang ya programnya lah

contoh program ros sederhana (python based)

```
#! /usr/bin/env python

import rospy #import library ros untuk python

rospy.init_node('ObiWan') #inisisasi kalau ini merupakan node bernama 'obiwan'
print("Help me Obi-Wan Kenobi, you're my only hope") #program print sederhana
```

Catatan penting :

- tiap python yang akan di run oleh ros, harus memilki node, atau gak nanti launch file nya bingung

- satu python hanya memiliki SATU node

ROS PACKAGE

Perlu kalian ketahui semua program pada ros pastinya akan berada pada satu package tertentu, hal ini dilakukan agar program kalian bisa berkomunikasi satu sama lain nya. Untuk membuat package kalian dapat melakukan Langkah Langkah sebagai berikut.

1. Masuk pada terminal linux kalian dan masukan command

   ```
   Roscore
   ```

   Apabila ros tidak terbuka kalian bisa melakukan penginstallan pada link berikut: http://wiki.ros.org/ROS/Installation

   Apabila ros telah menyala dengan normal buka tab terminal baru dan lanjutkan

2. Masuk ke dalam workspace yang telah kalian buat dan masuk ke folder src nya, pada tutorial ini command yang ku gunakan pada terminal adalah

   ```
   cd ~/abin/banyubramanta/workspace_sauvc/src
   ```

3. Saat kalian sudah masuk ke dalam folder src pada workspace kalian, kalian dapat memasukan command berikut pada terminal

```
Catkin_create_pkg package_satu rospy std_msgs
```

arti command diatas adalah sebagai berikut:

`catkin_create_pkg` : command dasar untuk membuat package pada workspace catkin

`package_satu` : nama package ros

`rospy` : ros python, memastikan python bisa digunakan pada package ini

`std_msgs` : standard message bawaan ros, ini akan berguna untuk memastikan package kalian bisa menggunakan ros message

Seharusnya, setelah melakukan command tersebut akan muncul pesan seperti ini pada terminal

Apabila pesan tersebut telah muncul, berarti kalian telah berhasil membuat package kalian

Perlu di ketahui segala compability seperti penggunaan Bahasa lain dan lain lain dapat di ubah nanti, jadi apabila kalian lupa untuk menambahkan `roscpp` atau `rospy` kalian tidak usah khawatir karena nanti bisa kalian tambahkan lagi pada package

Ros node

Rosnode merupakan program yang dijalankan oleh ros, anggaplah sebagai ID suatu program, untuk mengetahui program apa aja nih yang sedang berjalan di background ros kalian, kalian bisa menggunakan command

```
rosnode list
```

Dengan menggunakan command tersebut, kalian dapat mengetahui berbagai rosnode yang sedang berjalan di background, lalu apabila kalian ingin menetahui info lebih lanjut mengenai rosnode tersebut, dapat menggunakan command

```
rosnode info /nama rosnode
```

contoh:

```
rosnode info /obiwan
```

```
Node [/ObiWan]
Publications:
 * /rosout [rosgraph_msgs/Log]

Subscriptions:
 * /clock [rosgraph_msgs/Clock]

Services:
 * /ObiWan/set_logger_level
 * /ObiWan/get_loggers


contacting node http://ip-172-31-30-5:58680/ ...
Pid: 1215
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /clock
    * to: /gazebo (http://ip-172-31-30-5:46415/)
    * direction: inbound
    * transport: TCPROS
```

maka kalian akan mendapatkan output sebagai berikut:

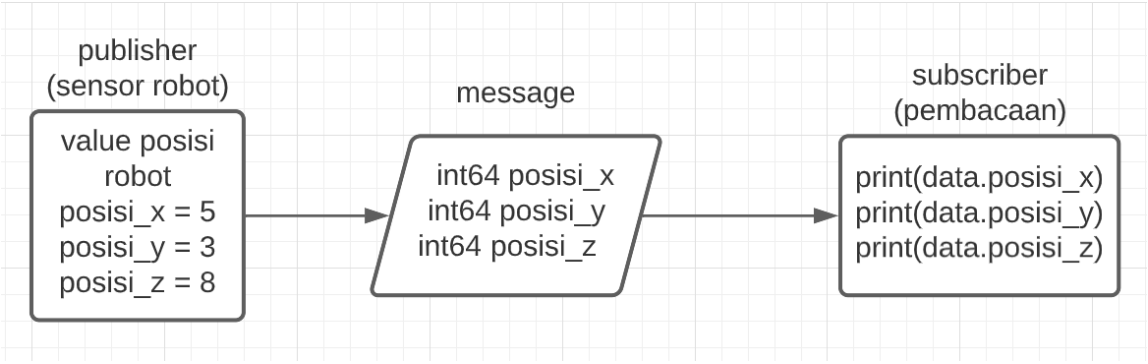
untuk detailnya, gausah khawatirin dulu, yang penting kalian berhasil buat mencari info mengenai node

Perlu kalian ketahui, node merupakan salah satu komponen terpenting dalam ros, hal ini dikarenakan node berfungsi sebagai semacam ID untuk program yang kalian jalankan dalam ros. Jadi mulai dari sekarang, setiap kali aku nge-refer node, berarti aku sedang ngomongin program.

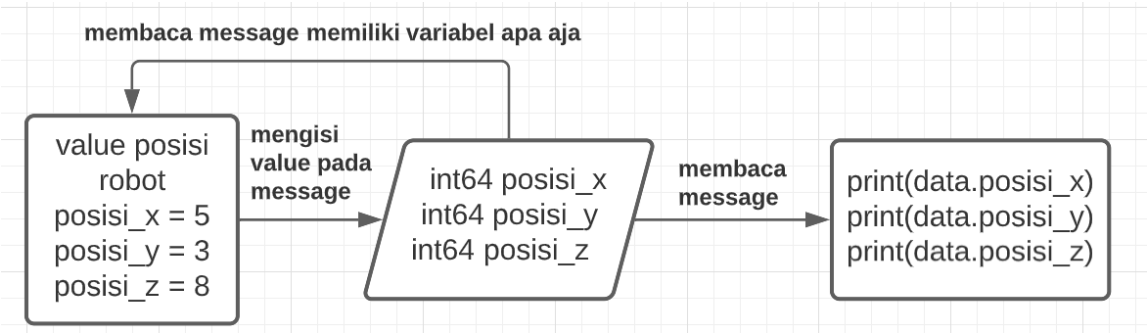
Important note : apa yang kutulis disini merupakan penjelasan oversimplified dari apa yang terjadi di lapangan, aku sarankan kalian membaca lebih lanjut lagi di ros wiki apabila kalian sudah tidak memiliki akses ke course the construct lagi.

ROS MESSAGE

Sebelum kita masuk pada ros topic, kalian harus memahami ap aitu ros message terlebih dahulu. Dasarnya ros message merupakan pesan yang digunakan pada ros topic untuk menyampaikan value pada program nya , simple nya akan di ilustrasikan pada gambar berikut.



Pada gambar diatas terdapat 3 komponen yaitu publisher, message dan subscriber, anggaplah pada gambar tersebut terjadi pembacaan sensor robot terhadap bidang vektor tiga dimensi yaitu sumbu x y dan z. pada program publisher, tiga value tersebut dipanggil dan dilemparkan Kembali pada message, jadi message tersebut memiliki isi dan dapat dibaca oleh subscriber, secara sederhana seperti berikut.



Perlu diketahui juga bahwa subscriber juga dapat mengisi ulang isi message , jadi untuk kemudahan diusahakan membuat satu message hanya diisi oleh satu publisher agar tidak terjadi tabrakan.

Pada dasarnya, message harus berada pada folder package dengan nama folder **msg**. hal ini merupakan hal yang **harus** karena apabila tidak, ros tidak dapat mendeteksi message yang telah dibuat. Isi ros message dapat berisi se sederhana mungkin seperti contoh berikut.

workspace_sauvc > src > package_satu > msg > kondisi.msg

```
1   string kondisi_robot
2   int64 persentase_baterai
```

Disini terdapat dua variable yaitu int64 yang dinamakan "persentase_baterai" dan string yang dinamakan "kondisi_robot".

Untuk mengetahui tipe variable yang disupport oleh message dapat dilihat pada link berikut : http://wiki.ros.org/msg

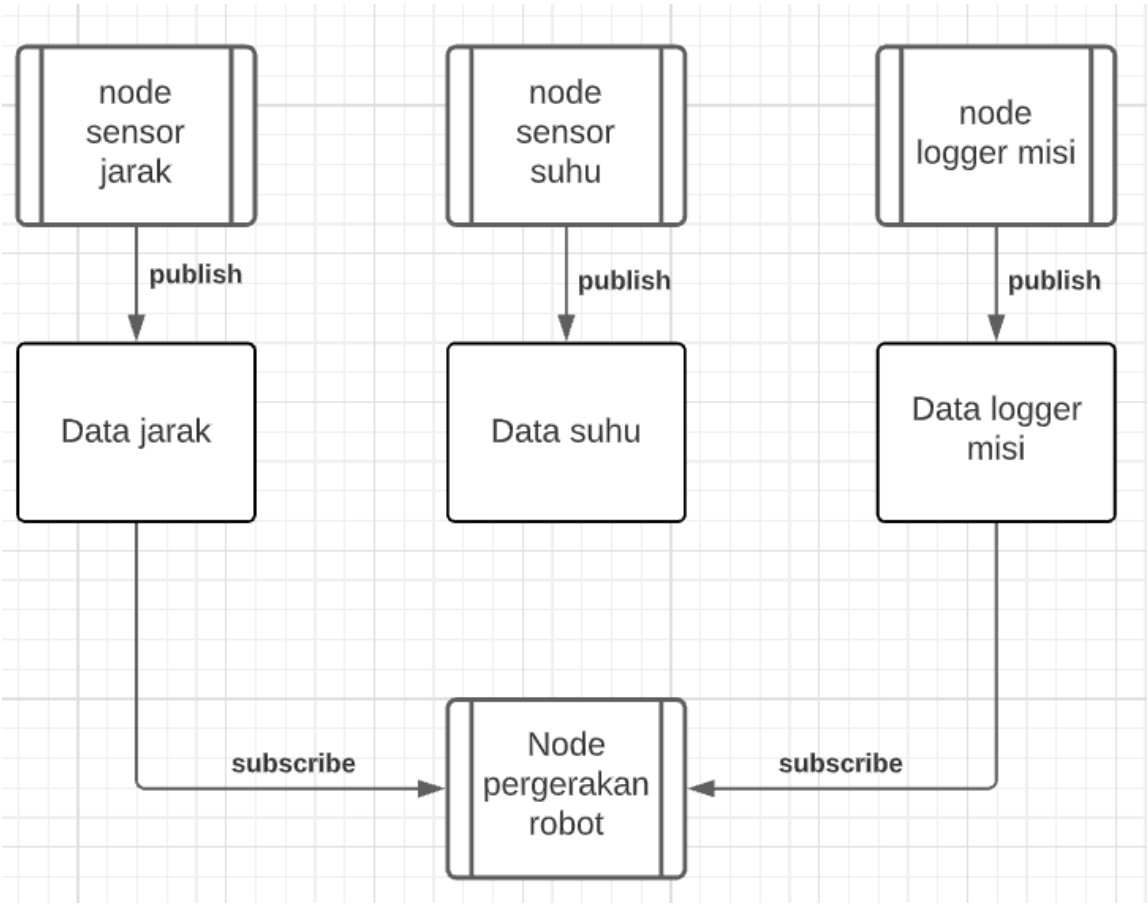
ROS TOPIC

Nah, sekarang aku akan membahas ros topic secara mendetail, sebelum itu, aku ingin mastiin kalau kalian sudah paham mengenai node secara mendetail. Apabila kalian sudah percaya diri tentang pengetahuan kalian dalam rosnode, kalian bisa melanjutkan lebih lanjut lagi.

Nah, sekarang apa sih topic itu ? topic secara sederhana merupakan sebuah program sekali pakai dalam ros yang dapat digunakan secara berkala dan sinkronus, apa artinya ? artinya untuk menjalankan suatu topic, kamu harus mengakhiri topic lain yang sedang berjalan.

Oke oke agak ngebingungin emang, jadi akan ku jelasin pelan pelan, ros topic memiliki dua sisi, yaitu **publisher** dan **subscriber**. Sesuai Namanya, publisher merupakan node yang menjadi kanal atau channel yang akan mempublish suatu informasi secara terus menerus pada kanal tersebut. Sementara subscriber adalah node yang membaca pesan yang dituliskan oleh subscriber pada kanal tersebut. Biasanya, tidak semua publisher memiliki subscriber, tetapi semua subscriber pasti memiliki publisher untuk di dengarkan.

Untuk mempermudah, kalian bisa lihat flowchart dibawah ini



Kira kira kalian bisa jelasin gak kaya gimana proses yang ada di flowchart diatas ? coba nanti bandingin dengan penjelasanku dibawah

Penjelasan

Flowchart diatas merupakan penyederhanaan pergerakan robot, bisa dilihat diatas ada 3 topic yang dibuat berdasarkan bacaan sensor berbeda yaitu jarak , suhu dan mission logger, nah 3 topic itu di publishkan pada 3 publisher berbeda tiap sensornya masing masing. Pada contoh kali ini aku mencontohkan ada subscriber yaitu node pergerakan robot. Misal pada pergerakan robot hanya memerlukan dua parameter yaitu sensor jarak dan data logger, maka node

pergerakan robot dapat hanya mendegarkan atau subscribe dua dari tiga publisher tersebut. Perlu diketahui juga bahwa satu publisher dapat di dengarkan oleh lebih dari satu subscriber juga.

Nah kalau kalian udah paham mengenai dasar dasar ros ayo coba mulai coding simple publisher dan subscriber kalian.

At this point, kalian harusnya udah mulai nginstall ros untuk memulai Latihan kalian, aku gak akan cover penginstallan ros karena udah disediain oleh wikiros.

PUBLISHER

Oke kita mulai dari publisher "sederhana" dulu, seperti yang sudah dituliskan tadi, bahwa publisher berfungsi sebagai pengirim pesan pada topic, jadi fungsi sebuah publisher adalah melemparkan pesan pada topic, isi pesan ini bisa berupa angka maupun huruf, aku belum mencoba gambar sih, tapi harusnya bisa. Anyway ini merupakan program sederhana publisher, perlu diketahui juga bahwa sebuah publisher juga bisa menjadi sebuah subscriber, intinya setiap program bisa dibuat menjadi penengah suatu sistem kompleks.

```python
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Secara sederhana, program tersebut dapat dibagi menjadi beberapa bagian berikut:

```
#!/usr/bin/env python #memastikan env python kalian telah (bisa diganti python3)
import rospy          #import library rospython
from std_msgs.msg import String #import nama string dalam file message yang berada di dalam package std_msgs
```

Bagian pertama merupakan bagian dimana kita memasukan berbagai library dan message yang diperlukan pada package kita kali ini.

```
user:~/workspace_sauvc$ rosmsg info std_msgs/String
string data
```

Berikut merupakan message pada file String.msg, dapat dilihat bahwa pada file msg tersebut terdapat variable string yang di namai "data"

kalau kamu di titik ini merasa bingung karena penamaan yang mirip mirip, percayalah kita sama. Tapi intinya adalah ada sebuah file msg Bernama String, yang berisi variable Bernama "data" dan variable tersebut berjenis string.

```
def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()
```

Selanjutnya ayo bahas inti dari publisher, pertama kita mulai dari

```
pub = rospy.Publisher('chatter, String , queue_size = 10)
```

nah fungsi dari line ini adalah untuk membuat variable pub pada program yang dimana variable ini akan memiliki beberapa parameter diantaranya adalah sebagai berikut

- `Rospy.publisher` : bawaan library rospy yang berfungsi untuk memberikan parameter bahwa variable ini merupakan publisher
- `'chatter'` : memberikan Id pada publisher agar bisa di call oleh subscriber, mirip seperti node tapi bukan node.
- `String` : memanggil String.msg yang telah di import dari std_msgs
- `Queue_size` : jumlah antrian maksimal (biar gak terlalu hectic)

Selanjutnya ada code

```
rospy.init_node('talker', anonymous=True)
```

pada line ini sebenernya sudah cukup jelasm bahwa ini merupakan line dimana seluruh file `subscriber.py` ini diberikan node yaitu Bernama 'talker' , mengenai `anonymous = true` , ini merupakan fail safe apabila pada ros kita terdapat dua program dengan nama node yang sama maka program yang paling barulah yang akan muncul sementara yang lama akan di berhentikan.

Selanjutnya ada code

```
rate = rospy.Rate(10) # 10hz
```

pada line ini sebenarnya aga ga penting ya, tapi ini merupakan rate pesan kita akan dikirimkan, ini dapat disesuaikan dengan jumlah queue size yang sudah di tetapkan agar tidak terjadi kekurangan atau kelebihan data variable.

Selanjutnya kalian bisa lihat ada block of code seperti ini, fungsi dari code ini adalah mengisi variable "data" pada string.msg yang telah di import tadi, akan coba gw breakdown satu persatu

```
while not rospy.is_shutdown():
    hello_str = "hello world %s" % rospy.get_time()
    rospy.loginfo(hello_str)
    pub.publish(hello_str)
    rate.sleep()
```

Bisa dilihat pada contoh terdapat parameter :

```
while not rospy.is_shutdown():
```

nah parameter ini berfungsi sebagai memastikan apabila rospy telah dimatikan , maka code tidak akan berjalan dan begitu juga sebaliknya, selama rospy berjalan maka code selanjutnya akan terus berjalan.

Selanjutnya terdapat variable

```
Hello_str = “hello world %s” %rospy.get_time()
```

- `Hello_str` : nama variable yang akan di publish
- `“hello world %s”` : isi dari variable, %s merupakan tambahan agar diberikan spasi (cmiiw)
- `% rospy.get_time` : mengambil waktu kapan variable telah di publish

Lalu ada line command `rospy.loginfo` nah fungsi dari line command ini adalah untuk mendapatkan log info dari publisher yang telah berjalan.

Yang terakhir adalah command `pub.publish(hello_str)`

- `Pub.publish` : merupakan pemanggilan variable “`pub`”
- `(hello_str)` : menamakan variable yang akan di publish

`Rate.sleep()` merupakan argument standar yang digunakan agar ros tertidur dalam rate yang telah di tetapkan agar dapat memenuhi loop.

Dan pada bagian terakhir terdapat code berikut

```
if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Disini terdapat argument standard dari python yaitu pengecekan __main__ , namun ditambahkan apabila terdapat interupsi ros maka program dapat di lewatkan.

Setelah semua program telah kalian tuliskan, kalian dapat melakukan hal standar setelah selesai ngoding, yaitu turu, maksudku compile.

Kalian bisa ngecompile package ros kalian dengan cara mundurkan terminal kalian ke workspace awal kalian lalu lakukan catkin build.

Ini wajib kalian lakukan secara berurutan atau kalian tidak dapat melakukan compile dengan benar.

```
Cd ~/abin/banyubramanta/workspace_sauvc

Catkin build

Source devel/setup.bash
```

Kalau kalian bingung apa yang ku tuliskan disini, kalian bisa baca beberapa notepad ku 😄

Setelah di compile kalian bisa coba run progam yang kalian buat menggunakan command berikut.

```
Rosrun package_satu publisher.py
```

SUBSCRIBER

Sekarang kita akan membahas apa itu ros subscriber. Secara mendasar ros subscriber adalah pendengar pada ros topic, subscriber akan mendengarkan message yang telah diisi oleh publisher, perlu di ketahui juga bahwa sebuah subscriber dapat menjadi sebuah publisher juga.

Sekarang aku akan menuliskan program subscriber "sederhana" agar kalian dapat memahami apa itu ros subscriber, perlu diketahui program ros subscriber yang ku tuliskan sekarang ini berhubungan dengan program ros publisher diatas jadi sangat ku rekomendasikan kalian membaca dan memahami ros publisher diatas dulu hingga paham sebelum masuk ke subscriber ini, apabila kalian sudah cukup paham mari kita mencoba masuk ke dalam ros subscriber.

```
1   #!/usr/bin/env python
2   import rospy
3   from std_msgs.msg import String
4
5   def callback(data):
6       rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)
7
8   def listener():
9
10
11      rospy.init_node('listener', anonymous=True)
12
13      rospy.Subscriber("chatter", String, callback)
14
15      # spin() simply keeps python from exiting until this node is stopped
16      rospy.spin()
17
18  if __name__ == '__main__':
19      listener()
```

Diatas merupakan program subscriber dari ros yang sederhana, seperti biasa aku akan nge breakdown satu persatu.

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String
```

Seperti biasa kita akan melakukan import library dan message yang akan kita masukan ke dalam subscriber kita, message yang kita masukan **wajib sama** dengan message yang kita gunakan pada publisher , apabila tidak maka message yang disampaikan oleh publisher tidak akan tersampaikan kepada subscriber, seperti perasaanku padanya ☹

Anyway, selanjutnya adalah

```
def callback(data):

    rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)
```

- `Rospy.loginfo` : rospy log selama program di run
- `Rospy.get_caller_id()` : mengambil id ros program subscriber ini
- `“I heard %s” , data.data` : menuliskan kata “I heard dan spasi, data.data atau isi string Bernama data yang di publish oleh publisher yang sedang di dengarkan

Selanjutnya ada yang merupakan inti dari subscriber yaitu

```
def listener():

    rospy.init_node('listener', anonymous=True) #inisiasi node

    rospy.Subscriber("chatter", String, callback)

    rospy.spin() #mengulang ulang seluruh proses
```

bagian terpenting dari code diatas adalah:

```
rospy.Subscriber("chatter", String, callback)
```

- `Rospy.subscriber` : menginisiasi subscribe
- `“chatter”` : nama publisher yang akan di subscribe
- `String` : nama message yang akan di subscribe
- `Callback` : fungsi callback diatas

Seperti biasa jangan lupa di compile programnya , dan sebelum menjalankan program subscriber jangan lupa di run juga program publishernya.