TECHNICAL REPORT UTS ROBOTICA MEMBUAT NODES PUBLISHER DAN SUBSCRIBER

FAISHAL ANWAR – TK44G7

1. Pertama tama hal yang harus kita lakukan adalah membuat workspace dalam hal ini adalah catkin_ws. Persyaratan pertama untuk bekerja dengan paket ROS adalah membuat ruang kerja catkin ROS. Setelah menginstal ROS, kita dapat membuat dan membangun ruang kerja catkin yang disebut catkin_ws:

```
mkdir -p ~/catkin_ws/src
```

Untuk mengkompilasi ruang kerja ini, kita harus menginisialisasi lingkungan ROS untuk mendapatkan akses ke fungsi-fungsi ROS:

```
source /opt/ros/noetic/setup.bash
```

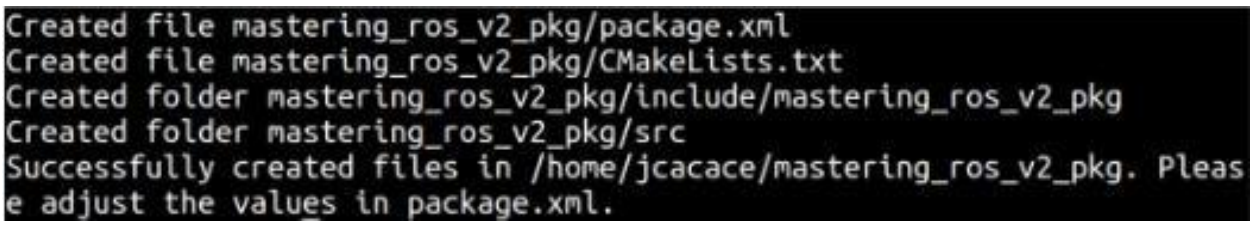
Beralih ke direktori src yang kita buat sebelumnya:

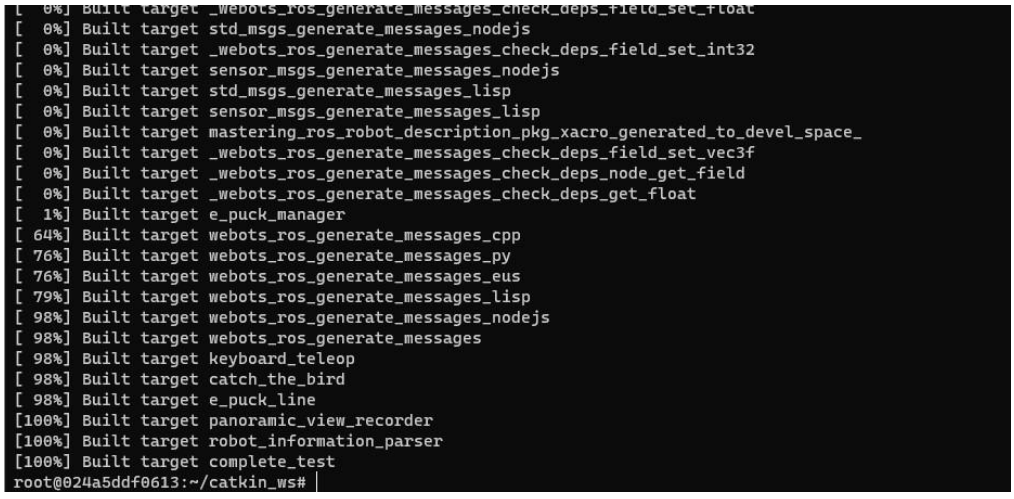
```
cd ~/catkin_ws/src
```

Setelah melakukan beberapa proses diatas, sekarang kita akan membuat catkin package, dengan cara;

```
catkin_create_pkg mastering_ros_demo_pkg roscpp std_msgs actionlib
actionlib_msgs
```

gambar dibawah merupakan hasil running perintah diatas;





2. Kemudian langkah selanjutnya yang perlu kita lakukan adalah membuat nodes, dalam kasus ini kita akan membuat 2 nodes, yang pertama adalah demo_topic_publisher.cpp dan demo_topic_subscriber.cpp

pertama tama kita buat nodesdemo_topic_publisher.cpp

```cpp
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

int main(int argc, char **argv) {
        ros::init(argc, argv,"demo_topic_publisher");
        ros::NodeHandle node_obj;
        ros::Publisher number_publisher = node_obj.advertise<std_msgs::Int32>("/numbers", 10);
        ros::Rate loop_rate(10);
        int number_count = 0;
        while ( ros::ok() ) {
                std_msgs::Int32 msg;
                msg.data = number_count;
                ROS_INFO("%d",msg.data);
                number_publisher.publish(msg);
                loop_rate.sleep();
                ++number_count;
        }
        return 0;
}
~
~
~
```

Kemudian kita buat nodes demo_topic_subscriber

```cpp
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

void number_callback(const std_msgs::Int32::ConstPtr& msg) {
        ROS_INFO("Received [%d]",msg->data);
}

int main(int argc, char **argv) {
        ros::init(argc, argv,"demo_topic_subscriber");
        ros::NodeHandle node_obj;
        ros::Subscriber number_subscriber = node_obj.subscribe("/numbers",10,number_callback);
        ros::spin();
        return 0;
}
~
~
~
~
~
~
~
~
```

Kemudian kita lakukan add executable pada CMakeList.txt, agar nodes yang sudah kita buat terbaca oleh program sebagai file executable.

```
  ${Boost_INCLUDE_DIRS}
)
#This will create executables of the nodes
add_executable(demo_topic_publisher src/demo_topic_publisher.cpp)
add_executable(demo_topic_subscriber src/demo_topic_subscriber.cpp)
#This will link executables to the appropriate libraries
add_executable(demo_msg_publisher src/demo_msg_publisher.cpp)
add_executable(demo_msg_subscriber src/demo_msg_subscriber.cpp)

add_dependencies(demo_msg_publisher mastering_ros_demo_pkg_generate_messages_cpp)
add_dependencies(demo_msg_subscriber mastering_ros_demo_pkg_generate_messages_cpp)
target_link_libraries(demo_msg_publisher ${catkin_LIBRARIES})
target_link_libraries(demo_msg_subscriber ${catkin_LIBRARIES})
target_link_libraries(demo_topic_publisher ${catkin_LIBRARIES})
target_link_libraries(demo_topic_subscriber ${catkin_LIBRARIES})

add_executable(demo_service_server src/demo_service_server.cpp)
add_executable(demo_service_client src/demo_service_client.cpp)
add_dependencies(demo_service_server mastering_ros_demo_pkg_generate_messages_cpp)
add_dependencies(demo_service_client mastering_ros_demo_pkg_generate_messages_cpp)
```

Kemudian lakukan running code dari kedua code nodes diatas setelah melakukan beberapa konfigurasi lain seperti yang disampaikan pada buku, berikut merupakan output code